Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий
Кафедра информатики, информационных технологий и методики обучения
информатике

ВИЗУАЛИЗАЦИЯ ПОИСКА ПУТЕЙ ОБХОДА ГРАФОВ КАК СРЕДСТВО РАЗВИТИЯ ИНЖЕНЕРНОГО МЫШЛЕНИЯ

Выпускная квалификационная работа
бакалавра по направлению подготовки
02.03.02 – Фундаментальная информатика и информационные технологии

Исполнитель: студент группы Б-41
Института математики, информатики и ИТ
Брагина Е. Ю.

Руководитель: д.п.н., зав. кафедрой
ИИТиМОИ
Лапенок М. В.

Работа допущена к защите
«____»_____ 2016 г.
Зав. кафедрой _____

Екатеринбург – 2016

Реферат

Брагина Е.Ю. ВИЗУАЛИЗАЦИЯ ПУТЕЙ ОБХОДА ГРАФОВ КАК СРЕДСТВО РАЗВИТИЯ ИНЖЕНЕРНОГО МЫШЛЕНИЯ, выпускная квалификационная работа: 61 стр., рис. 20, табл. 2, библ. 30 назв., приложений 4.

Ключевые слова: ЭЛЕКТРОННЫЙ ОБРАЗОВАТЕЛЬНЫЙ РЕСУРС, ИНЖЕНЕРНОЕ МЫШЛЕНИЕ, ГРАФ, ПОИСК ПУТИ.

Объект разработки – электронный образовательный ресурс, направленный на развитие инженерного мышления у студентов, посредством визуализации поиска путей обхода графов.

В работе рассмотрены методы развития инженерного мышления у студентов технических специальностей, показана целесообразность изучения студентами путей обхода графа. Обоснована необходимость создания электронно-образовательного ресурса, который будет включать совокупность графовых задач, направленных на развитие инженерного мышления студентов, и обеспечит визуализацию решения данных задач. Описаны результаты проектирования и программной реализации электронного образовательного ресурса «Поиск путей обхода графов».

Ресурс разработан на языке объектно-ориентированного программирования C#, посредством функциональных возможностей среды разработки Visual Studio 2010.

Оглавление

Введение

Устойчивое развитие современного общества глобальной коммуникации и информатизации, функционирование экономики, требуют поиска путей для опережающей разработки новых инновационных технологий производства. Решение этой задачи невозможно без наличия достаточного количества инженерных кадров высокой квалификации. Это обусловливает необходимость развития у студентов, осваивающих точные и естественные науки, инженерного мышления, т.е. мыслительной деятельности, ориентированной на преобразование человеком окружающего мира с целью удобства и оптимизации жизнедеятельности самих индивидов. Результатом такого мышления являются инновационные идеи, внедрение которых обеспечивает повышение эффективности производства и опережающую разработку инновационных технологий.

Иллюстрацией вышесказанного является существующий в промышленном секторе Свердловской области дефицит компетентных и высококвалифицированных инженерных кадров по многим специальностям.

Из-за проблем с процессом инвестирования в человеческий капитал, этот дефицит стал носить затяжной характер, причем недостаток квалифицированных специалистов просматривается на всех стадиях воспроизводства жизненного цикла промышленной продукции, от технической подготовки производства, вплоть до эксплуатирования оборудования.

По данным правительства Свердловской области в 2015 году промышленные предприятия были укомплектованы инженерами, конструкторами и технологами лишь на 70 процентов, ощущается также дефицит специалистов в области информационных технологий. Данная проблема усугубляется из-за образовавшегося разрыва между требованиями работодателей к умениям сотрудников и образовательными стандартами, увеличилось несоответствие между спросом и предложением компетентных в

инженерной сфере кадров. Помимо этого ситуация ухудшается еще и по причине того, что средний возраст высококвалифицированного инженерно-технического персонала составляет 53 года и выше [27, с. 3].

В связи с этим можно утверждать, что существует необходимость в некотором комплексе мероприятий, который будет способствовать повышению интереса обучающихся к изучению предметов по инженерным специальностям и улучшению качества подготовки специалистов в системе высшего образования данной области. Например, для этого следует сделать акцент на наглядной демонстрации поставленной задачи в графическом виде, что облегчит понимание материала и натолкнёт студента на возможные решения такой задачи. При этом следует учитывать интеллектуальные особенности студентов. Восприятие новой информации будет более качественным, действенным и продуктивным при использовании подобной методики. [29, с. 22] А именно графовые модели как раз подойдут в качестве средства для наглядного представления задачи, которую нужно решить.

Объект исследования: методы и средства развития инженерного мышления у студентов.

Предмет исследования: средства информационных технологий, способствующие развитию инженерного мышления у студентов.

Цель: разработать электронный образовательный ресурс для визуализации алгоритмов, основанных на графах, при решении инженерных и математических задач.

Для реализации цели были поставлены следующие задачи:

1. проанализировать научно-методическую литературу в области методов и средств развития инженерного мышления с целью выбора наиболее эффективных и действенных;
2. выделить типовые инженерные задачи, исследовать возможность применения графовых алгоритмов для их решения и составить

комплект учебных задач, способствующих развитию инженерного мышления у студентов;

3. выполнить анализ научно-методической литературы в области создания и использования электронных образовательных ресурсов в учебном процессе, требований, предъявляемых к ним и методам оценки их педагогико-эргономических качеств;
4. разработать с использованием средств компьютерной визуализации электронный образовательный ресурс, направленный на овладение студентами на теоретическом и практическом уровнях методами решения графовых задач;
5. подготовить сопроводительную документацию.

Глава 1. Теоретическое обоснование применения графовых задач для развития инженерного мышления

1.1. Инженерное мышление и методы его развития

Современное общество с каждым днем становится все более зависимым от инновационных технологий, и именно поэтому в настоящее время пристальное внимание уделяется такой области человеческого интеллекта, как инженерное мышление. Хотя вопросы, связанные с инженерным мышлением интересовали ученых и педагогов еще на этапе формирования инженерного образования [23], сейчас же повышение качества инженерно-технического образования, в соответствии с требованиями современного производства , является одной из важнейших задач инженерной педагогики.

Значимость формирования инженерного корпуса подчеркивается созданием в Свердловской области комплексной программы "Уральская инженерная школа", утвержденной губернатором Свердловской области Е.В. Куйвашевым, которую планируется реализовать в период с 2015 по 2034 годы. Большое внимание в этой программе уделяется содействию производственным предприятиям в их усилиях по подготовке и реализации потенциала высококвалифицированных инженерных кадров с целью повышения эффективности производства [27, с. 3]. Для успешной реализации программы и осуществления, поставленных в ней целей необходимо сформировать у студентов компетенции в области инженерной деятельности, что обусловливает необходимость формирования и развития у будущих кадров инженерного мышления.

По мнению ряда авторов [23,с. 23-24, 25,с. 48-55], инженерный склад ума представляет собой особый вид мышления, который является совокупностью таких основных видов мышления, как наглядно-образное, наглядно-действенное, теоретическое, практическое, техническое, логическое,

конструктивное, творческое и исследовательское. В процессе практической деятельности человека осуществляется систематический переход от одного вида мышления к другому.

Так за возможность решить проблемную ситуацию, по не стандартному алгоритму отвечает творческое мышление. Оно позволяет креативно подойти к решению проблемы, используя такие качества, как находчивость, изобретательность, сообразительность, предприимчивость, адаптивность, а это в свою очередь способствует возникновению принципиально нового, возможно инновационного решения.

Логическое мышление, позволяет при помощи доказательств и рассуждений получить аргументированный вывод из имеющихся предпосылок, путем оперирования логическими понятиями и конструкциями

Наглядно-образное мышление отвечает за решение поставленных задач посредством представления ситуации и манипулирования с образами ее комплектующих, без выполнения реальных практических действий с ними.

В свою очередь наглядно-действенное мышление характеризуется тем, что поставленная задача решается с помощью реальной, физической модификации состояния и испытания свойств объекта.

Теоретическое мышление основывается на поиске решения без осуществления практических действий, то есть человек решает задачу только мысленно, пользуясь готовыми знаниями, представленными в виде умозаключений, понятий, суждений, убеждений и соображений.

Практическое мышление основывается на суждениях и умозаключениях сформированных при решении практических задач и направлено на преобразование окружающей действительности с помощью определения целей, составления планов, восприятия и манипулирования настоящими материальными предметами.

Исследовательское мышление позволяет самостоятельно осваивать и оригинально перестраивать новые способы деятельности, выявлять необычное

и новое в задаче, сопоставлять с уже известными категориями задач, подкреплять доказательствами свои действия, а так же давать обоснование полученным результатам и на основе этого приходить к умозаключению

Техническое мышление позволяет анализировать состав, устройство, структуру и понимать принцип работы неизвестных ранее технических объектов, читать чертежи и схемы.

Конструктивное мышление определяется отсутствием эмоциональных, ситуативных идей при принятии решений, оно позволяет выстраивать модель решения поставленной проблемы или задачи, основываясь на анализе и использовании изученных теоретических знаний и полученных ранее практических умений.

Требование современных работодателей, заключающееся в том, чтобы сотрудник не только обладал знаниями в предметной области и мог реализовать поставленную задачу, но и умел правильно презентовать свои возможности, послужило причиной в необходимости наличия экономического мышления у инженеров. Такое мышление позволяет осуществлять самостоятельный анализ качества процесса и результата деятельности с позиций требований рынка.

Основываясь на вышесказанном, под инженерным мышлением будем понимать мыслительную деятельность, ориентированную на преобразование человеком окружающего мира с целью удобства и оптимизации жизнедеятельности самих индивидов [23, с. 23]. А формируется и проявляется такое мышление при решении задач инженерного типа, что позволяет специалистам быстро, точно, креативно и с наименьшими затратами решать поставленные задачи. Результатом инженерного мышления являются инновационные идеи, внедрение которых обеспечивает повышение эффективности производства и опережающую разработку инновационных технологий, что является важной причиной для выработки инженерного мышления у студентов, осваивающих точные и естественные науки.

В работах современных исследователей (Л.В Занфирова, А.Н. Стась, О.Н. Прусских) рассмотрены методы формирования инженерного мышления. Так, Л.В. Занфирова [15, с. 16-20] обосновала систему методов, обеспечивающую комплексную, последовательную, регулярную и результативную работу по формированию инженерного мышления. Система включает следующие методы:

- формирование образного компонента инженерного мышления, то есть выработку у студентов умений, которые позволят переходить от образа к понятию и в обратном порядке, выходить за пределы увиденного, тем самым раскрывать процесс ретроспективных событий или их прогнозирование в будущем.
- формирование действенного компонента инженерного мышления, а именно, наработки практических и интеллектуальных навыков, позволяющих решать технические, производственные, графические, технологические и конструкторские задачи, а так же развития таких мыслительных процессов (операций), которые позволят работать с техническими объектами при решении соответствующих задач;
- формирование понятийного компонента инженерного мышления, то есть создание такой системы технических знаний, которая имела бы эффективную целостную структуру;
- формирование оперативности инженерного мышления, то есть умения применять полученные знания в ситуациях близких к реальным профессиональным при решении задач, в условиях ограниченного времени;
- формирование интегративности инженерного мышления, обусловливающего умение использовать понятия и средства, усвоенные/освоенные при изучении различных наук, умение рассматривать объект исследования с разных ракурсов и прогнозировать последствия принятого решения;

- формирование творческого уровня инженерного мышления, то есть таких творческих умений инженера как: обнаружение и формулировка технических проблем; генерация множества всевозможных идей; видение шаблонной ситуации с новых ракурсов; планирование и проведение исследовательских действий для проверки своего варианта решения технической задачи; графическое выражение собственного варианта технического объекта; анализ представленного варианта решения задачи; обоснование полученных результатов и эффективности предложенного решения;
- формирование рефлексивного уровня инженерного мышления, обусловливающего осознание студентом сути процессов мышления, способов формирования личности, результатов становления собственной личности (в аспекте формирования профессиональных качеств), достигнутых в процессе учебной деятельности.

На основании вышеприведенных материалов, можно сделать вывод о том, что система научно обоснованных методов формирования инженерного мышления, выстроенная в соответствии с его определением, структурой и качественными характеристиками, является обширной и емкой основой для создания конкретных программ, которые будут способствовать развитию инженерного мышление у студентов по определенным техническим и естественнонаучным дисциплинам.

1.2. Графовые задачи как средство развития инженерного мышления

А.Н. Стась и О.Н. Прусских [24, с. 166-167] на основании сформулированного В.Р. Поповой определения инженерного мышления и выделения Л.В. Занфировой системы методов, доказывают, что хороших результатов в развитии инженерного мышления можно достичь в процессе решения графовых задач, основанных на алгоритмах (традиционных и

эвристических) поиска пути обхода графов. Авторы убедительно показывают, что методы теории графов завоевали всеобщее признание, как средство решения математических, естественнонаучных и технических задач благодаря простоте вычислений и наглядности представления результатов [1, 10]. Уникальность такого подхода к решению задач определяется тем, что мониторинг промежуточных результатов обучения, анализ накапливаемых учебных достижений позволяют преподавателю достаточно просто оценить успешность обучения каждого студента, и. следовательно, просто и удобно регулировать сам процесс обучения. В теории графов используются такие средства описания условия задачи, как таблицы, модели, графы, которые являются наглядным, естественным и понятным для человека способом представления сути задачи (поскольку вследствие образного мышления человека графическое представление материала воспринимается легче, чем текст). А простота восприятия и является одним из важнейших критериев эффективного развития инженерного мышления [28].

В процессе изучения теории графов, студенты знакомятся с основными понятиями этого раздела дискретной математики и базовыми алгоритмами, в результате чего у них формируется представление о методах решения различных задач на основе графов. В свою очередь реализация алгоритмов поставленных задач на некотором языке программирования (то есть создание программ) является достаточно сложным процессом, который состоит из множества этапов, таких как постановка задачи, ее алгоритмизация, проектирование и отладка программы.

Так теория графов стала мощным средством исследования и решения многих задач, которые рождались при изучении больших и сложных систем. Например, для специалистов по информационным системам и вычислительной технике теория графов - это наглядный, простой, удобный, а так же убедительный язык выражения понятий из этой области. Многие результаты

теории графов имеют непосредственную связь с задачами, с которыми таким специалистам приходится сталкиваться в своей работе.

Любую логическую схему, например, можно представить в виде ориентированного графа. Такие модели помогают оценить быстродействие схемы. А если блок-схема некоторого алгоритма представлена вероятностным графом, то это позволяет оценить временные характеристики алгоритма, затраты процессорного времени, трудоемкость и другие. Так называемым графом "дерево" можно показать практически любую структуру организации или предприятия.

Кроме того теория графов получила широкое применение при исследовании проблемы оптимизации, которая возникает когда требуется модификация большой технической или программной системы для улучшения ее работы с целью большей эффективности. Это значит, что многие прикладные задачи оптимизации могут быть сформулированы в форме той или иной задачи оптимизации на графах.

Существует достаточно большое число типовых задач оптимизации на графах, из которых можно выделить основные: задача нахождения оптимальных покрывающих деревьев; задача нахождения кратчайшего пути в графе; задача нахождения критического пути в сетевом графе; задача нахождения максимального потока в графе. Для каждой из перечисленных задач дается в соответствие математическая постановка задачи в форме булевского или целочисленного программирования. А так же существуют специальные алгоритмы их решения, которые учитывают специфические особенности постановки этих задач.

Так связь между задачами программирования и задачами теории графов была представлена доктором физико-математических наук В.А. Евстигнеевым [11, с. 295] (см. Рис.1).

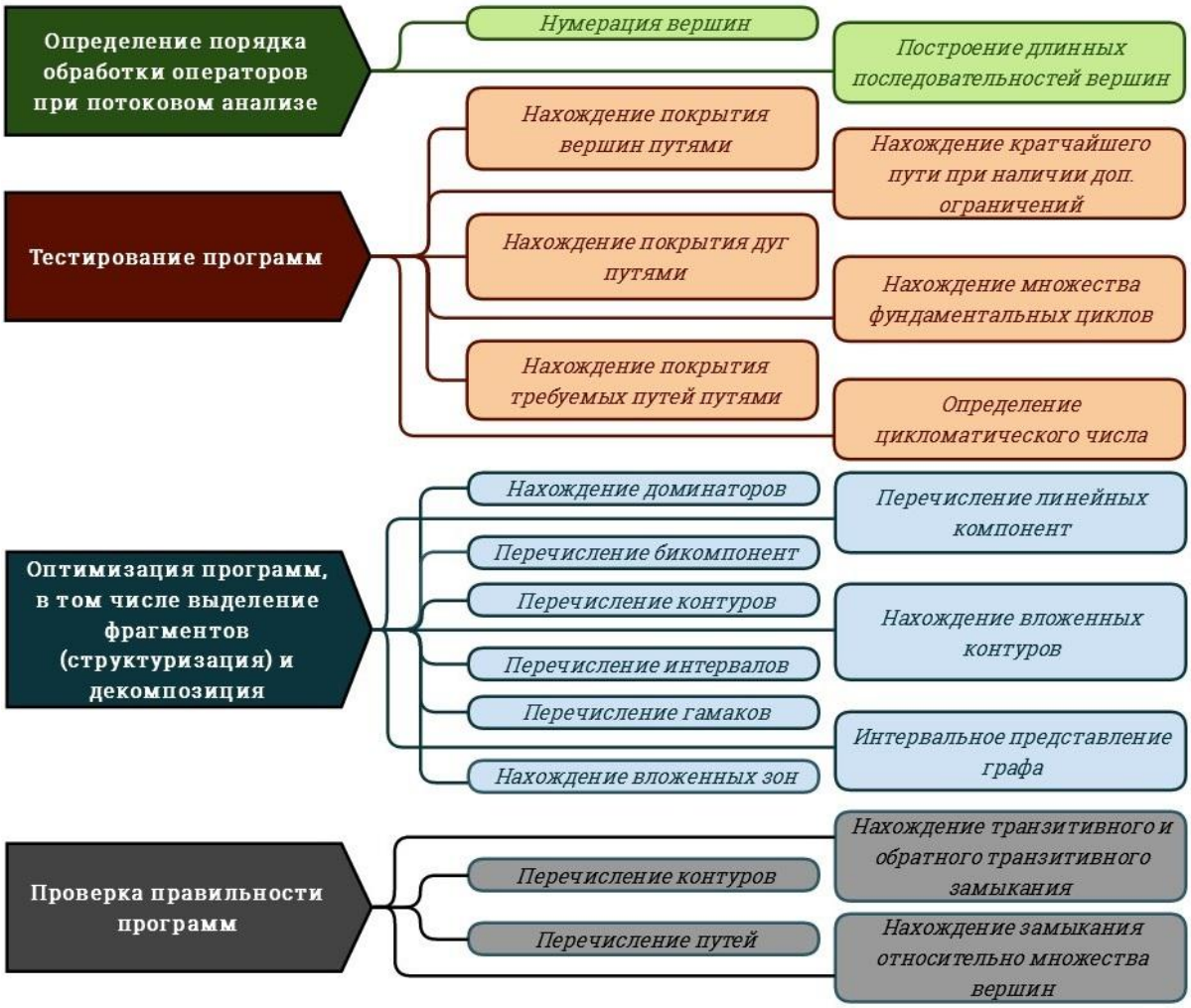


Рисунок 1. Задачи программирования и сопоставленные им задачи теории графов

Но для того, что бы выделить типовые инженерные задачи и исследовать возможность применения графовых алгоритмов для их решения, требуется для начала дать определение инженерной задаче. Так как инженерное дело-это решение инженерных задач, очень важно с самого начала точно представить себе, что же такое инженерная задача.

Такая задача возникает всякий раз, когда нужно перейти от одного состояния к другому. Два состояния могут быть двумя точками в пространстве, расстояние между которыми должно быть измерено. Задачей может быть переправа с одного берега реки на другой, переезд из города в город, перелет с планеты на планету и тому подобное. Задача часто возникает тогда, когда нужно перейти от одного физического состояния к другому, например хлеб → гренки. У любой задачи есть начальные условия, которые называют состоянием

A, или входом, а то состояние, которого нужно достичь, называют состоянием B, или выходом (см. Рис.2).

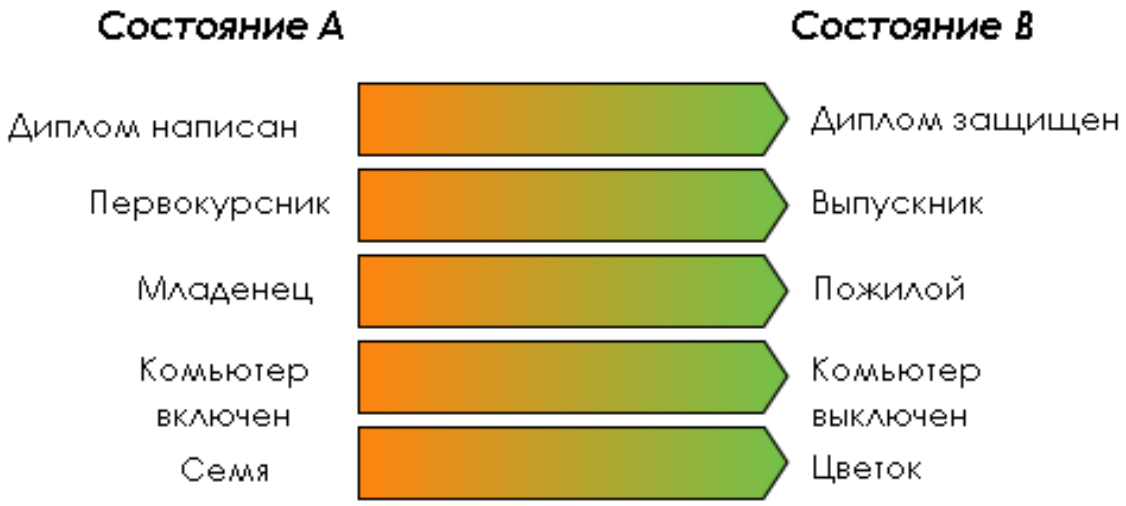


Рисунок 2. Примеры представления задач с помощью состояний A к B.

Большинство задач такого рода имеет огромное число решений, то есть - различных способов перехода из одного состояния в другое. Так, например, есть много видов транспорта и много возможных маршрутов между двумя пунктами на земле. Однако не все эти варианты следует принять к рассмотрению, но, тем не менее, они существуют. В случае же, если нет различных способов достижения требуемого результата, то нет и инженерной задачи. Точно так же если все возможные решения одинаково хороши, то инженерной задачи не существует.

Инженерная задача - это нечто большее, чем нахождение одного решения; она требует нахождения предпочтительного метода достижения желаемого результата[2, с. 6-7]. Так, например, для большинства пассажиров небезразлична цена, скорость, степень безопасности, комфорт и надежность, присущие различным видам транспорта. Основной признак, по которому одно решение выбирается из многих возможных, называется критерием.

При переходе из состояния A в состояние B часто существуют определенные средства, применение которых неизбежно, потому что они определены теми, с чьим авторитетом инженер должен считаться. Допустим, например, что нужна переправа через реку и что для этого должен быть

обязательно выбран мост, а не паром; или гренки должны быть поджарены только с помощью электрической плитки. Средства, которые должны быть обязательно применены при решении задачи, называются ограничениями.

Таким образом, задача существует тогда, когда, требуется перейти от одного состояния к другому, если существует более чем одно возможное решение и если все возможные решении не очевидны.

Из вышесказанного следует, что теория графов может быть использована в качестве базы для формирования и дальнейшего развития инженерного мышления, являясь естественным средством объяснения сложных ситуаций на интуитивном уровне [11]. Кроме того можно говорить о том, что для решения типовых инженерных задач возможно применение графовых алгоритмов, что облегчит процесс нахождения верного ответа, сделают его более очевидным, удобным и понятным.

В свою очередь одной из главнейших основополагающих задач теории графов является задача о кратчайшем пути. На сегодняшний день известно о большом количестве разнообразных алгоритмов для её решения. Основываясь на множестве различных областей применения данной задачи, можно говорить о том, что она играет важнейшую роль. Например, прокладывание линий электропередач и сетей автодорог, авиалиний и железнодорожных путей, в геолокационных и картографических сервисах, в криптографии. Кроме того нельзя не отметить, что большое значение решение такой задачи имеет при организации маршрутизации в сетях связи и представляет собой важную составляющую процесса разработки и обучения искусственных нейронных сетей, который представляет собой определяющую тенденцию структурного подхода по исследованию возможности создания естественного интеллекта с помощью компьютерных алгоритмов. Без поиска кратчайшего пути не обойтись и при решении разного вида задач, таких как: транспортные, сетевого планирования и управления, дискретные задачи динамического программирования.

Прежде чем говорить о самой задаче поиска кратчайшего пути и описании алгоритмов ее решения, следует дать определения используемым терминам [16, с. 5, 6, 7].

Граф представляет собой конечную совокупность объектов, которые соединяются между собой связями. Объектами, из которых состоит граф, являются вершины, а связи представлены в виде ребер.

Вершина является точкой, в которой могут сходиться или из которой могут выходить ребра.

Ребро является линией, которая соединяет пару смежных вершин графа.

Если вершины графа соединены ребром, то они называются смежными, в противном случае - несмежные.

Ориентированный граф представляет собой граф, в котором все ребра имеют направление. В таком графе вершины будут называться узлами, а ребра - дугами.

Под **взвешенным графом** будем понимать граф каждому ребру которого поставлена в соответствие дополнительная информацию, которая называется вес ребра. В большинстве случаев весом ребра является некоторое вещественное число, в таком случае вес будет называться длиной ребра.

Используя, термины, которые были представлены выше, можно дать определение задачи поиска кратчайшего пути. Данная задача заключается в нахождении такого пути между вершинами на графе, который будет иметь минимальную сумму весов ребер или состоять из минимального количества вершин. При этом за вес каждого из ребер можно принимать не только расстояние, но и временные интервалы, пропускная способность, убытки, расходы, то есть любую другую метрику, которая по мере прохождения пути будет линейно накапливаться и которую в конечном итоге нужно привести к минимальному значению.

Задача о нахождении кратчайшего пути может использоваться как для ориентированных и взвешенных графов, так и для не ориентированных и не

взвешенных [18]. При этом постановка задачи может принимать следующий вид:

- Задача о кратчайшем пути из заданного пункта назначения.

Находится путь, который берет свое начало в каждой из вершин графа, а заканчивается в конкретной заданной конечной вершине. При этом данную формулировку задачи можно привести к такому виду, что будут находиться всевозможные минимальные пути из пункта отправления до всех остальных вершин, при условии, что заданна единая исходная вершина.

- Задача о кратчайшем пути между заданной парой вершин.

Заключается в нахождении кратчайшего пути из вершины отправления в вершину назначения, каждая из которых будут заданы изначально.

- Задача о кратчайшем пути между всеми парами вершин.

Находится кратчайший путь из каждой в каждую из существующих вершин.

Существует четыре базовых алгоритма, которые позволяют решить данную задачу.

Алгоритм Дейкстры.

Данный алгоритм характеризуется тем, что позволяет находить оптимальные маршруты и их длину между заданной исходной вершиной и всеми остальными вершинами графа. Недостаток алгоритма заключается в том, что при наличии в графе ребер/дуг с отрицательными весами, его работа будет некорректной [3, 4]. Следовательно, использовать данный алгоритм при решении задач, в которых предусматривается наличие отрицательных, то есть убыточных маршрутов, не представляется возможным. Кроме того при работе данный алгоритм не предусматривает петель, однако он не накладывает ограничения на вид графов для которых будет использоваться.

Принцип работы алгоритма Дейкстры заключается в следующем:

1) Создаются два массива, один из которых будет хранить информацию о том, посещена вершина или нет и второй, в который по ходу

работы программы будут заноситься длины найденных кратчайших путей. При этом изначально все вершины отмечаются как не посещенные, а в массив с путями, которые будут найдены, заносятся такие большие числа, которые заведомо будут больше длины любого из найденных потенциальных путей;

2) Длина пути для вершины, которая является стартовой, заносится в массив изначально и будет равной 0;
3) Находятся такие смежные вершины, которые являются соседями для стартовой вершины, то есть имеют ребра, соединяющие их с исходной вершиной.
4) Для каждой из найденных вершин высчитывается стоимость маршрута от стартовой вершины, полученные длины заносятся в соответствующий массив, при этом каждое из новых значений сравнивается с уже находящимися в массиве данными и заносится только в том случае, если новая длина меньше предыдущей;
5) Вершина, для которой на предыдущем шаге были найдены смежные вершины, отмечается как посещенная;
6) Из смежных с исходной вершин, выбирается новая вершина, путь до которой минимален из стартовой вершины;
7) Для выбранной вершины выполняются все предыдущие шаги, так же как и для стартовой.
8) Алгоритм выполняется до тех пор, пока все вершины, которые могут быть достигнуты из стартовой вершины не будут отмечены как посещенные.

Алгоритм Беллмана-Форда.

Задачу о кратчайшем пути из фиксированной вершины до каждой из остальных вершин графа (при этом вес дуг может быть отрицательным) позволяет решить алгоритм Беллмана-Форда. Достоинство данного алгоритма заключается в том, что он позволяет определить наличие отрицательного цикла

в графе, то есть такого цикла, в котором веса входящих в него дуг в сумме представляют отрицательное число. Однако в силу того, что каждый очередной проход по отрицательному циклу, лишь улучшает значение минимального пути, который требуется найти. Таким образом, неограниченное число улучшений делает невозможным нахождение одного конкретного значения, являющегося оптимальным. Следовательно, алгоритм Беллмана-Форда не применим к графам, в которых существуют отрицательные циклы, однако в модифицированном виде он позволяет определить их наличие [7, 8]. Кроме того алгоритм Беллмана-Форда не используется для графов у которых отсутствуют веса.

Решение задачи нахождения кратчайшего пути, с помощью данного алгоритма, основывается на методе динамического программирования, то есть разбиении одной основой задачи на типовые подзадачи меньшего размера, найти для них оптимальные решения и использовать найденные ответы для достижения желаемого результата, то есть решения исходной задачи. Определение кратчайшего пути для одной отдельно взятой вершины, до любой другой в графе и является решением для каждой из таких подзадач.

Последовательность действий при выполнении алгоритма заключается в следующем:

1) Создается массив, который будет содержать значения кратчайших путей из одной вершины в другую. В начале работы алгоритма всем элементам данного массива присваиваются некоторые большие значения, которые заведомо не могут быть больше любой из найденных в дальнейшем длин путей.
2) Так как путь из стартовой вершины в саму себя равен 0, это значение записывается в созданный массив.
3) С помощью двух циклов на каждой фазе просматриваются все ребра графа, и алгоритм пытается произвести релаксацию вдоль каждого ребра заданной стоимости. Это означает, что на каждом шаге

осуществляется проверка позволяющая улучшить значение в массиве длин и если длина найденного пути будет меньше длины записанной ранее, то это значение перезаписывается.

4) Таким образом, для конкретной, заданной изначально вершины находятся пути с минимальной длиной до всех остальных вершин.

Алгоритм Флойда-Уоршелла.

Алгоритм Флойда-Уоршелла позволяет вычислить значения кратчайших путей из каждой вершины графа до каждой, используя метод динамического программирования [5, 6]. Данный алгоритм может использоваться только на взвешенных графах, с положительными и отрицательными весами ребер. При этом в графах должны отсутствовать отрицательные циклы.

Действия при реализации алгоритма заключаются в следующем:

1) Формируется матрица смежности, значения которой в ходе работы алгоритма будут меняться. Каждому элементу данной матрицы присвоен вес соответствующего ребра. Так же создается матрица, в каждом элементе которой будет записан номер вершины, из которой можно попасть в данную вершину. Данная матрица понадобится для того что бы восстановить сам путь.
2) По главной диагонали матрица заполняется 0, всем остальным элементам имеющим значение 0, присваивается некоторое большое число.
3) Далее выполняется ключевая часть алгоритма, которая состоит из трех циклов, то есть перебираются все пары вершин. Каждая из пар проверяется с помощью условия, которое позволяет определить существует ли путь из первой вершины во вторую, через третью вершину и меньше ли этот путь того пути, который был найден ранее. В случае выполнения условия пересчитываются длины кратчайших путей между данными вершинами.

4) В результате после выполнения алгоритма в матрице расстояний для каждой из вершин графа будет записана длина кратчайшего пути между ними, либо большое число, если пути между этими вершинами не существует.
5) Восстановить сам кратчайший путь можно по матрице элементами, которой теперь являются такие номера вершин, из которых можно достичь данной вершины идя по кратчайшему пути.

Алгоритм «Фронта волны».

Данный алгоритм применяется для как для неориентированных, так и для ориентированных графов, но не допускает использование на графах с весами. В основе алгоритма «Фронта волны» лежит метод обхода графа в ширину. В общем случае алгоритм позволяет находить только длину кратчайшего пути из одной вершины в другую, но с помощью его модификации можно достичь восстановления самого пути [13].

Процесс нахождения кратчайшего пути в графе с помощью данного алгоритма состоит в том, чтобы каждой из вершин графа присвоить такой индекс, который будет равен длине кратчайшего пути из начальной вершины в данную. При этом присваивание индексов производится в следующем порядке:

1) Стартовой вершине присваивается индекс равный 0.
2) Всем вершинам, смежным со стартовой присваивается индекс 1. Вершинам, смежным с вершинами, которые имеют индекс 1, присваивается индекс, на единицу больше, то есть 2.
3) Выполнение алгоритма заканчивается в том, случае если вершине, до которой искали кратчайший путь, будет присвоен индекс.
4) Чтобы восстановить сам кратчайший путь требуется двигаться из конечной вершины в направлении убывания индексов.

Описанный алгоритм принято называть волновым или алгоритмом «фронта волны», по причине того что процесс расстановки отметок напоминает распространение возмущения, то есть волны, которое возникает в вершине и

движется со скоростью одно ребро в единицу времени. Вершины, значения отметок которых одинаковы, являются фронтом волны.

На основании описания алгоритмов, были составлены таблицы, в одной из которых представлены ограничения на вид графов для каждого из алгоритмов (см. Таблица 1), а в другой – постановка задачи о нахождении кратчайших путей, для решения которой будет применяться определенный алгоритм (см. Таблица 2).

Таблица 1.

Ограничения для алгоритмов по виду графов

Название алгоритма / *Ограничения*	**«Фронта волны»**	**Дейкстры**	**Беллмана-Форда**	**Флойда-Уоршелла**
Не ориентированные	✓	✓	✓	✓
Ориентированные	✓	✓	✓	✓
Не взвешенные	✓	✓	✗	✗
Взвешенные	✗	✓	✓	✓

Таблица 2.

Задачи решаемые алгоритмами

Название алгоритма / *Задача*	**Фронта волны**	**Дейкстры**	**Беллмана-Форда**	**Флойда-Уоршелла**
О кратчайшем пути из заданного пункта отправления	✗	✓	✗	✗
О кратчайшем пути между заданной парой вершин.	✓	✗	✓	✗
О кратчайшем пути между всеми парами вершин.	✗	✗	✗	✓

Следовательно, можно сделать вывод о том, что каждый из алгоритмов будет решать определенную задачу о кратчайших путях на графе и использоваться для нахождения путей на графах определенного вида.

Таким образом, в результате соотнесения методов формирования инженерного мышления, содержания предметной области «Информатика и информационные технологии» и обоснования использования графовых задач, в качестве метода развития инженерного мышления нами была выявлена необходимость в создании электронного образовательного ресурса. Так же были установлены способы, выделены средства, определены направления деятельности по развитию инженерного мышления у студентов.

На основании вышеприведенных фактов можно говорить о том, что именно электронный образовательный ресурс, содержание которого отражает теоретическое обоснование методов обхода графов, а форма представления информации обеспечивает визуализацию этих методов, может применяться в качестве средства формирования и развития инженерного мышления [24].

Кроме того на основываясь на данном определении инженерных задач и описании алгоритмов поиска путей обхода графа был составлен комплект учебных задач, который направлен на овладение студентами на теоретическом и практическом уровнях методами решения инженерных задач при помощи алгоритмов поиска путей в графе (см. **Ошибка! Источник ссылки не найден.**). Назначение данного комплекса состоит в том, чтобы:

- способствовать формированию инженерного мышления;
- требовать от студентов умений мысленно преобразовать и обрабатывать полученный материал, использовать инструменты, представленные в созданном электронном образовательном ресурсе с максимальной эффективностью, а так же находить оптимальный вариант ответа из нескольких, полученных в результате самостоятельно проделанной работы;

- включать задачи разного уровня сложности, предполагать несколько вариантов решений и иметь изменяющиеся параметры, что позволит студентам усовершенствовать приобретенные навыки и обеспечит переход на более высокие уровни развитости инженерного мышления.

Таким образом, решение представленного комплекса задач будет выполнять свою главную функцию, то есть способствовать развитию инженерного мышления у студентов.

1.3. Техническое задание

Техническое задание на разработку электронного образовательного ресурса, составленное на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

1. Общие сведения.

1.1. Организация-заказчик: Уральский Государственный Педагогический Университет (УРГПУ).

1.2. Продукт разработки: электронный образовательный ресурс для визуализации алгоритмов поиска путей обхода графов (далее ЭОР «Поиск путей обхода графов»).

1.3. Продукт имеет ряд назначений:

- овладение студентами на теоретическом и практическом уровнях методами решения графовых задач;
- использование в качестве средства формирования и развития инженерного мышления;
- организация интерактивной работы обучающихся с помощью возможностей современной компьютерной техники.

1.4. Сроки начала и окончания работ:

- сентябрь - составление содержательного плана ВКР;

- сентябрь - выбор объекта, обоснование актуальности темы, формулировка технического задания на разработку, выбор технологических решений и методов;
- октябрь-февраль - анализ текущего состояния вопроса, выбор и обоснование путей решения, написание Главы 1;
- октябрь-февраль – разработка ЭОР «Поиск путей обхода графов».
- февраль - промежуточная аттестация на заседании кафедры;
- февраль-март - апробация разработанных материалов в учебном процессе.
- март - подготовка публикации по теме «Визуализация путей обхода графов как средство развития инженерного мышления»;
- апрель-май - написание Главы 2 и заключительной части.
- май-июнь - предварительная защита на заседании кафедры, окончательное оформление работы, подготовка доклада и презентации к защите;
- июнь - сдача ВКР в Государственную аттестационную комиссию.

2. Характеристика области применения продукта.
   2.1. Процессы и структуры, в которых предполагается использование продукта разработки: в учебном процессе в качестве средства, способствующего изучению материалов по основам теории графов и овладению студентами методами решения графовых задач.
   2.2. Характеристика персонала: для использования данного продукта от пользователя не требуется специальных навыков, кроме знания основ работы с компьютером.
3. Требования к продукту разработки.

3.1. В целом к готовому продукту предъявляются следующие требования: круглосуточная возможность работы с ресурсом, не требующая доступа к сети Интернет, кроме того от пользователя не должно требоваться специальных технических навыков, знания технологий или программных продуктов, помимо основных навыков работы с персональным компьютером

3.2. Аппаратные требования - персональный компьютер, со следующими характеристиками:

- процессор Intel® Pentium® 4, AMD Athlon™ 64 или AMD Opteron™;
- оперативная память: 1Гб;
- свободное место на жестком диске около 5 Мбайт;
- монитор с разрешением 1024 x 768.;
- устройства ввода: компьютерная мышь, клавиатура.

3.3. Указание системного программного обеспечения:

- операционная система Microsoft® Windows® 8, Microsoft® Windows® 7 (32- или 64-разрядная версия), Windows Vista® (32- или 64-разрядная версия) или Windows ® XP (32-разрядная версия) с последними пакетами обновления
- платформа Net Framework версии 2 или новее;
- текстовый редактор позволяющий работать с файлами формата *.txt и/или *.doc;
- стандартная (или любая другая) программа для просмотра изображений;
- программа для работы с файлами *.pdf.

3.4. Для реализации продукта использовалось следующее программное обеспечение:

- интегрированная среда разработки программного обеспечения Microsoft Visual Studio 2010;
- текстовый процессор Microsoft Office Word предназначенный для создания, просмотра и редактирования материалов руководства пользователя и материалов по основам теории графов;
- текстовый редактор позволяющий работать с файлами формата *.txt и/или *.doc;
- стандартная программа для просмотра изображений.

3.5. Форматы входных и выходных данных:

- входными данными является информация о графе, его количественные характеристики (тип графа, количество вершин, ребер, их вес);
- выходные данные представляют собой файл с изображением готового, полноценного графа и файл со списком кратчайших путей обхода данного графа по выбранному алгоритму.

3.6. Источники данных и порядок их ввода, вывода и хранения в ЭОР «Поиск путей обхода графов»:

- с помощью инструментов, расположенных на специальной панели, в области визуализации строится граф, как совокупность вершин и соединяющих их ребер;
- выбирается алгоритм поиска пути из представленного списка, для этого нужно нажать левой кнопкой мыши на нужный алгоритм, после чего он автоматически отметится, как выбранный;
- в специальные текстовые поля ввести требуемую информацию (про начальную и конечную вершины) и нажать на кнопку поиска путей.

- в специальной области выводится список найденных путей, каждый из которых можно показать на графе, для этого следует мышкой выбрать в списке нужный путь;
- чтобы сохранить полученную информацию в файлы, нужно выбрать соответствующие пункты меню, после чего список путей обхода графа сохранится в текстовый файл, а изображение графа в графический файл изображения.

3.7. Порядок взаимодействия с другими системами, возможности обмена информацией:

- При копировании. Посредством буфера промежуточного хранения Windows и ЭОР может обмениваться данными с другими прикладными программами. Чтобы передать такие данные как найденный путь из ЭОР, нужно выделив мышкой нужный путь воспользоваться командой Копировать из меню Правка.
- При сохранении списка путей. Список полученных путей по выбранному алгоритму можно сохранить в файл формата *.txt и/или *.doc, который впоследствии можно открыть с помощью любого текстового редактора позволяющего работать с файлами данного формата, например, такого как стандартное приложение «Блокнот».
- При сохранении изображения графа. Изображение построенного графа можно в любой момент сохранить в графический файл, который можно будет открыть стандартной (или любой другой) программой для просмотра изображений.
- При вызове справочной информации по работе с программой. В этом случае используется файл справки и файл с основной информацией по теории графов и по алгоритмам поиска путей обхода графов, которые были созданы в программе HTML Help

Workshop, и открываются при выборе соответствующих подпунктов меню.

3.8. В качестве мер по защите информации в программе реализованы запреты на ввод некорректной информации или информации неверного типа, а так же запрет на редактирование таких частей программы, в которых отображается выходная информация, изменение которой может привести к некорректной работе программы.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса:

- интерфейс должен быть простым в использовании, интуитивно понятным и комфортным для пользователя;
- должны отсутствовать элементы, не соответствующие используемому шаблону, по причине того что они выделяются из множества однотипно оформленных элементов;
- управляющие элементы должны быть строго выровненными;
- объекты, отображенные на экране, должны быть созданы, основываясь на образах и свойствах объектов из окружающего мира;
- такая характеристика как абсолютная симметричность затрудняет видение и восприятие информации с экрана;
- все элементы одной категории должны быть одинакового цвета и размера, что позволит однозначно определить их принадлежность.
- все компоненты приложения должны быть сдержанных цветов, в одной цветовой гамме и дополнять друг друга, так как слишком агрессивные, насыщенные или не сочетаемые цвета, могут отвлекать внимание пользователя или вводить его в заблуждение, а, следовательно, создавать трудности в работе.

4.2. Размещение информации на экране, дизайн экрана:

- блоки на главной форме ЭОР «Поиск путей обхода графов» должны быть расположены таким образом, чтобы в первую очередь пользователь обращал внимание на область визуализации графа, так как она является основным блоком программы, в котором происходит процесс прорисовки графа, поэтому она должна располагаться в центре.
- остальные блоки, находящиеся по бокам будут равноправными, поэтому последовательность их чтения не очевидна, что может привести к рассеиванию внимания, но большая часть пользователей, сначала сфокусирует внимание на левом блоке и только после этого на верхнем и правом.

Такой макет представления информации не допускает сосредоточения внимания пользователя только на одной стороне экрана, потому что логические блоки информации продуманно размещены на форме (см. Рис.3).

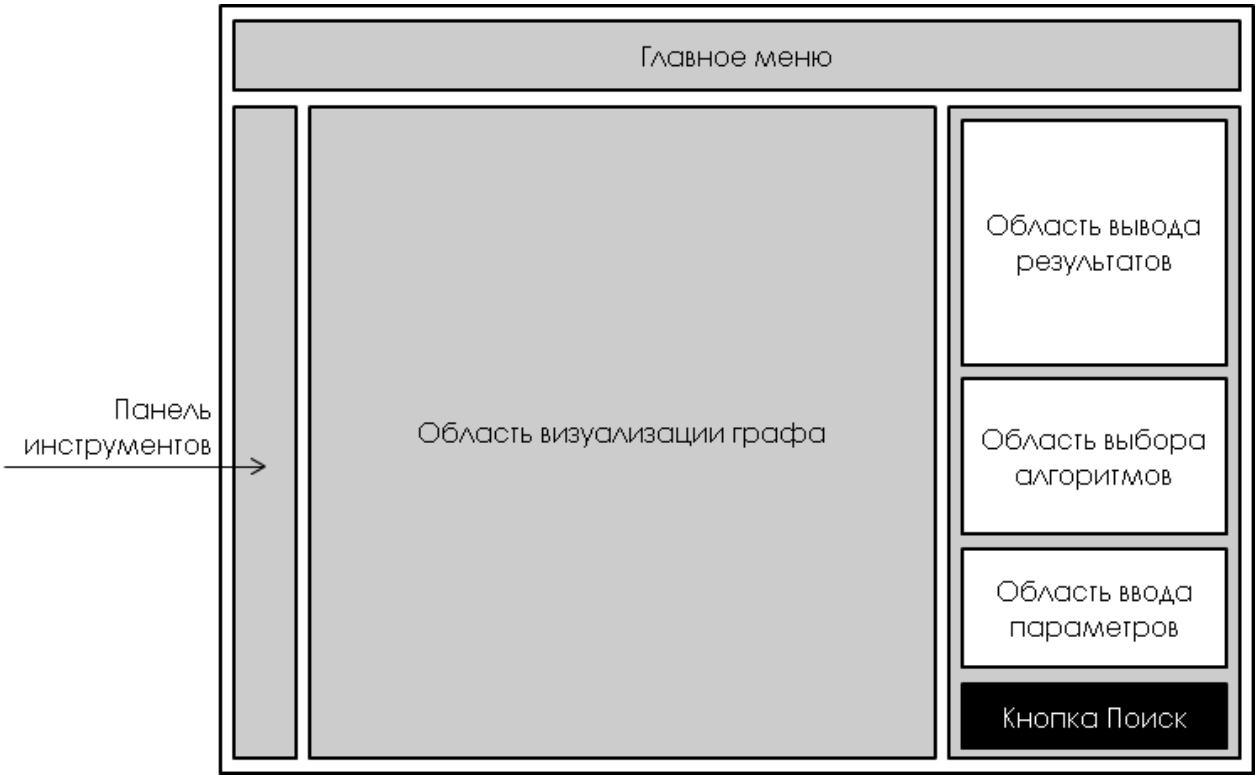


Рисунок 3. Макет ЭОР

4.3. Особенности ввода информации пользователем, представление выходных данных:

Существуют требования, используемые применительно к такому аспекту оценки электронных образовательных ресурсов, как особенности ввода информации пользователем. Они устанавливают что:

- формат ввода данных должен быть удобным и комфортным для пользователя;
- должны присутствовать процедуры проверки достоверности и корректности введенных данных;
- должна быть организована совместимость с другими структурами данных.

Требования, относящиеся к выходным данным, в основном касаются представления их в правильной терминологической форме. Данные требования характеризуют непосредственные информационные нужды пользователей в нескольких аспектах и содержат ряд пунктов:

- выходные данные должны быть представлены в читабельном, удобном и понятном для пользователя виде;
- пользователю должна быть предоставлена возможность для удобного доступа к данным;
- должна быть предусмотрена функция для пользовательского отбора данных.

5. Требования к документированию.

5.1. Перечень сопроводительной документации:

- Техническое задание. Является главным документом, сопровождающим ЭОР «Поиск путей обхода графов» и отражает все характеристики приложения и требования к нему.
- Руководство пользователя. Документ сопроводительной документации к данному ресурсу, предназначенный для

помощи пользователю в работе с ресурсом и для возможности разрешения вопросов, возникающих при эксплуатации ЭОР.

5.2. Требования к содержанию отдельных документов:

- Техническое задание - это исходный документ для проектирования информационной системы. В его состав должны входить технические требования, предъявляемые к продукту и исходные данные для разработки; в нем указывается информация о назначении объекта, области его применения, стадиях разработки документации, её состав, а так же сроки выполнения.
- Руководство пользователя должно содержать информацию о назначении электронного образовательного ресурса, областях его применения, аппаратно-программные требования к ресурсу, описание всех его компонентов, а главное - подробные инструкции для помощи пользователю в использовании системы и для ее полноценной и корректной работы.

Глава 2. Электронный образовательный ресурс для визуализации алгоритмов поиска путей обхода графа

2.1. Содержательные и эргономические требования, предъявляемые к электронным образовательным ресурсам

Приведенные в главе 1.2 преимущества представления сложных структур и процессов с помощью графов становятся еще более ощутимыми при наличии качественных средств их визуализации. Что в свою очередь объясняет растущий в настоящее время интерес к методам и системам визуальной обработки графов и графовых моделей.

XXI век - это век информационно-коммуникационных технологий, поэтому в современном образовании применяются самые различные образовательные ресурсы, позволяющие в полной мере использовать возможности современной компьютерной техники для организации интерактивной работы обучающихся. В настоящее время информационные технологии имеют огромный диапазон возможностей для повышения качества системы образования в целом и учебного процесса в частности.

Одним из дидактических средств информационной образовательной среды, обладающих значительным развивающим потенциалом, является мультимедиа, контент в котором одновременно представлена информация в различных формах - текстовая, аудиальная, графическая, видеоинформация и анимированная компьютерная графика. Для представления такой информации используются электронные устройства. Но современная организация образовательного процесса ориентирована во многом на самостоятельное использование студентом учебных ресурсов и рассчитана на то, что сам студент управляет процессом обучения, а не является пассивным и безынициативным. Поэтому возникает необходимость в таком компоненте информационной образовательной среды, который будет ориентирован на реализацию

образовательного процесса с помощью информационно-коммуникационных технологий, с применением новых методов и форм обучения основанных на самостоятельной работе студентов. Такой результат достигается с помощью использования электронных образовательных ресурсов

Возможность использования электронных образовательных ресурсов в большей мере обуславливается такими качествами объекта, которые позволяют использовать его с дидактическими целями в образовательном процессе, такими как [21]:

1. **Интерактивность** (обеспечивает динамическое управление учебными объектами, и дает возможность вмешательства в процессы электронного образовательного ресурса, отвечает за возникновение ответных реакций на действия студента и за автоматизацию различных видов учебных работ)
2. **Коммуникативность** (обеспечивает оперативную передачу учебных материалов студенту, позволяет преподавателю регулировать процесс обучения с помощью использования средств мониторинга учебных достижений и автоматизированной оценки сформированности компетенций, отвечает за быстрый и удобный доступ к ресурсу.)
3. **Наглядность** (преподнесение информации в понятном для студентов виде посредством применения различных форм представления учебного материала).

Следовательно, использование электронных образовательных ресурсов в образовательном процессе обладает рядом существенных преимуществ,

- осуществление самостоятельной, продуктивной, нацеленной на результат деятельности студентов
- осуществление мониторинга промежуточных результатов обучения, анализа накапливаемых учебных достижений, а, следовательно,

индивидуальной поддержки и оценки успешности обучения преподавателем каждого из студентов.

- осуществление, с помощью инструментов информационно-коммуникационных технологий, таких видов образовательной деятельности как: электронное обучение, мобильное обучение, сетевое обучение, автономное обучение, смешанное обучение, совместное обучение;
- осуществление автоматизации нетворческих, рутинных операций и поиска необходимой информации, что влечет за собой улучшение эффективности учебной деятельности и повышение интереса студентов к изучаемому предмету.

На основании национальных стандартов Российской Федерации разрабатываемые электронные образовательные ресурсы должны соответствовать ряду требований [19, 20].

Требования, касающиеся функционального назначения электронного образовательного ресурса:

- электронный образовательный ресурс должен быть направлен на организацию образовательного процесса с использованием новых методов и форм обучения, которые основаны на применении информационно-коммуникационных технологий;
- предметное содержание ЭОР, а так же его структура, и метаданные должны соответствовать назначению ЭОР в образовательном процессе и требованиям, которые предопределенны спецификой деятельности в информационно-образовательной среде;
- функциональная структура ЭОР должна основываться на специфике уровней образования и изучаемых дисциплин (предметов), а так же соответствовать назначению ЭОР в образовательном процессе;

- представление структуры ЭОР может быть основано на блоках учебного материала, которые являются совместно используемыми объектами содержания.

Требования к метаданным:

- для обеспечения эффективного поиска и объединения элементов контента при создании новых ЭОР следует использовать такие структурированные данные, которые предназначены для описания характеристик образовательных ресурсов, то есть метаданные.
- для обеспечения интероперабельности метаданных в различных информационно-образовательных средах следует использовать базовую информационную модель метаданных. Для описания основных характеристик ЭОР такая модель содержит упорядоченный набор элементов;
- для регламентирования процессов создания, представления, обработки, хранения и использования метаданных ЭОР, следует разработать профиль метаданных, представляющий собой совокупность стандартов и нормативно-технических документов.

ЭОР является продуктом, создаваемым на основе знаний о предметной области с использованием педагогических методов, дидактических подходов и средств информационно-коммуникационных технологий. Поэтому существует список требований, касающихся комплекса отличительных свойств, которыми должен обладать ЭОР, определяющих присущие ЭОР характеристики качества. Данный список может быть условно разделен на три основные категории:

- отличительные свойства, характеризующие соответствие структуры и содержания ЭОР требованиям федеральных образовательных стандартов, образовательных программ, нормативных и учебно-методических документов;

- отличительные свойства, характеризующие ЭОР с точки зрения педагогических, дидактических и психологических аспектов его использования в образовательном процессе;
- отличительные свойства, характеризующие ЭОР как продукт информационно-коммуникационных технологий с учетом специфики его использования в информационно-образовательной среде.

Кроме того, для обеспечения повышение качества и эффективности процессов создания и эксплуатации информационно-образовательных средств, ЭОР должны подлежать классификации, которая представлена в стандарте[], устанавливающем принципы рубрикации электронных образовательных ресурсов.

Качество созданного ЭОР оцениваются на основании характеристик, представленных в ГОСТ Р ИСО/МЭК 9126-93 [20]:

1) **Функциональные возможности** (оценивается функции и свойства в зависимости от установленных требований.)
2) **Надежность**(оценивается свойство программного обеспечения сохранять свой уровень качества функционирования за определенный период времени при заданных условиях.)
3) **Практичность** (оценивается объем работ, которые требуются для использования ресурса определенным кругом пользователей.)
4) **Эффективность** (оценивается соотношение между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при заданных условиях.)
5) **Сопровождаемость** (оценивается набор атрибутов, отвечающих за объем работ, который потребуются для проведения конкретных модификаций)
6) **Мобильность** (оценивается способность программного обеспечения быть перенесенным из одного окружения в другое.)

В результате анализа требований было определено, что для разработки и создания ЭОР «Поиск путей обхода графа» необходимо реализовать следующие задачи:

1) Выделить, обосновать, сформировать совокупность теоретических материалов, содержащих системно и последовательно изложенные материалы по основам теории графов, которые будут способствовать формированию инженерного мышления у студентов.
2) Разработать с использованием средств компьютерной визуализации электронный образовательный ресурс, направленный на овладение студентами на теоретическом и практическом уровнях методами решения графовых задач.

Таким образом, решение поставленных выше задач будет способствовать созданию условий для формирования у студентов компетенции в области инженерной деятельности и развитию у них инженерного мышления

2.2. Выбор технологии проектирования и реализации электронного образовательного ресурса

На сегодняшний день существует всего лишь несколько методологий программирования, каждая из которых представляет собой комплекс специальных понятий, концепций, принципов, методов и образов, который формирует определенный стиль разработки, отладки и сопровождения программ. Каждая конкретная методология программирования лежит в основе определенной технологии программирования. Главная цель любой методологии состоит в изучении и внедрении, таких способов проектирования и реализации программ, которые смогут упростить решение процесса оптимизации и исправления ошибок программного обеспечения после сдачи его в эксплуатацию. С помощью методологии процесс создания программного продукта становится четко отрегулированным, упорядоченным и

организованным, вследствие чего труд разработчика становится эффективнее более оперативным и, а значит и гораздо плодотворнее.

Для создания ЭОР, который будет визуализировать построение графа, а так же находить и обозначать кратчайшие пути на графе, была выбрана объектно-ориентированная методология разработки. Данный выбор основывается на том, что при построении любого графа следует опираться на его количественные характеристики, такие как вершины и ребра, каждая из которых имеет свои свойства и методы. Такое представление напрямую соответствует понятию класса, то есть некоторой структуры, описывающей совокупность однотипных объектов и их поведение. Именно потому, что понятие класса и операций, производимых над объектами этого класса, лежит в основе объектно-ориентированного программирования, данную методологию рационально использовать при разработке данного ЭОР.

Кроме того, лежащая в основе объектно-ориентированной методологии программирования, структура в виде классов обладает следующим списком достоинств:

- благодаря классам существует возможность разделить громоздкую и сложную программу на несколько разграниченных частей, что представляет собой практичный инструмент, позволяющий разработчику абстрагироваться и не обращать внимания на незначительные детали реализации;
- местоположение кода и данных таково, что позволяет усовершенствовать визуализацию и сделать более удобным процесс сопровождения (поддержки) программного продукта.
- такой важнейший механизм объектно-ориентированного программирования как инкапсуляция обеспечивает защиту конфиденциальных данных от несанкционированного вмешательства извне.

- одно из самых главных преимуществ, это возможность создания расширяемых систем, которые характеризуются работой с новыми компонентами, без внесения в саму систему каких либо изменений.

Для разработки современного программного обеспечения в настоящее время существует множество различных технологий и платформ разработки. Так например в последнее время большое распространение получил язык объектно-ориентированного программирования C# и платформа NET. Данный язык предназначается для разработки различных безопасных и мощных пользовательских приложений с удобным интерфейсом. C# является типобезопасным, гибким и выразительным языком, имеющим ряд существенных преимуществ:

- прост в изучении, особенно для тех, кто знаком с такими языками программирования как: C, C++, Java или Delphi. Так разработчики знающие любой из этих языков и обладающие знаниями об основных принципах программирования, с легкостью смогут освоить C# и в результате эффективной работы получить полноценный готовый продукт;
- кроме обычных приложений Windows позволяет создавать XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения баз данных и не только.
- предоставляет в распоряжение развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств для практичной и комфортной разработки приложений.
- в некоторых аспектах является более простым, чем C++ и обеспечивает более мощные возможности, например такие как, прямой доступ к памяти, чего в свою очередь не позволяет сделать Java, кроме того C# упрощает разработку компонентов программного обеспечения, с помощью ряда инновационных конструкций;

- обеспечивает поддержку универсальных методов и типов, что обуславливает высокий уровень безопасности и производительности;
- предоставляет возможность обеспечить взаимодействие с другим программным обеспечение Windows;
- процесс построения C# по сравнению с C и C++ прост и является более гибким, чем в Java. Отсутствуют отдельные файлы заголовка, а методы и типы не требуют объявления в определенном порядке. В исходном файле C# может быть определено любое число классов, структур, интерфейсов и событий.

Программа на языке C# выполняется в среде .NET Framework. Данная среда является интегрированным компонентом Windows, содержащим виртуальную систему выполнения (среда CLR) и унифицированный набор библиотек классов [26]. Ключевой особенностью .NET Framework является взаимодействие между языками. Это обусловлено тем, что код создаваемый компилятором на промежуточном языке (IL), соответствует спецификации CTS и, следовательно, может взаимодействовать с кодом, создаваемым версиями языков Visual Basic, Visual C++, Visual J# платформы .NET Framework и еще порядка более чем 20 CTS-совместимых языков. Таким образом, в одной сборке может быть несколько модулей, написанных на разных языках платформы .NET Framework, а типы могут ссылаться друг на друга, как будто они были написаны на одном языке (см. Рис.4)..

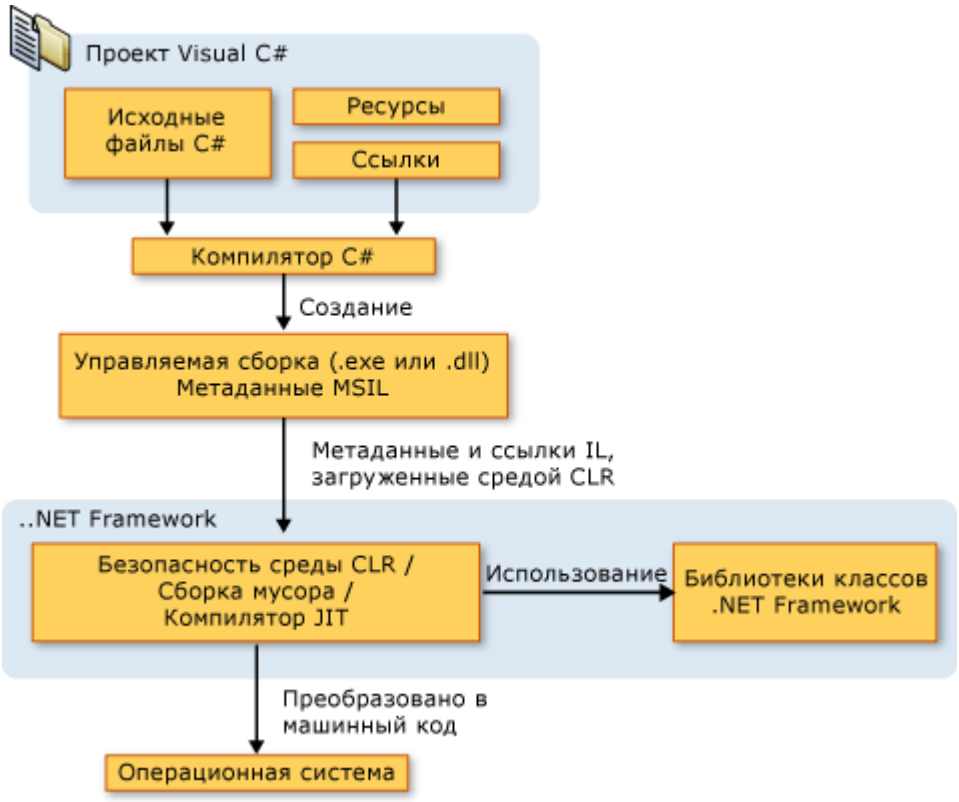


Рисунок 4. Схема стадий по созданию, компиляции и сборки проекта Visual C#

На основании вышеуказанных преимуществ, языком для разработки электронного образовательного ресурса по визуализации поиска путей обхода графов, был выбран объектно-ориентированный язык программирования C#, на платформе NET Framework. В качестве интегрированной среды разработки программного обеспечения был выбран продукт Visual Studio компании Microsoft, который предоставляет возможность разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

2.3. Структурно-функциональная модель и описание созданного электронного образовательного ресурса

В начале разработки ЭОР следует иметь четкое представление о том, какими функциональными возможностями он будет обладать, а так же каким образом будут организованы процессы взаимодействия между его элементами. Таким образом, первым пунктом в процессе создания ЭОР стало его проектирование, то есть создание функциональной модели, которая позволит в подробностях описать морфологию ресурса, то есть определить его

составляющие и связи между ними, а так же четко сформулировать список функций которые должен в полной мере выполнять ресурс.

Главной особенностью таких моделей является то, что они с помощью графических элементов, которые в совокупности напоминают блок схемы, позволяют отображать полный состав ресурса, выполняемые им функции, а так же порядок действий, направленных на достижение поставленной задачи или ряда задач.

Функциональные модели обладают рядом существенных преимуществ, а их возможности позволяют[22]:

- визуализировать работу и детализировать состав сложных, больших по объему систем, что упрощает их понимание и облегчает работу с ними;
- создать точное описание моделируемого объекта, даже в том случае, когда заранее неизвестно какова будет реализация данного объекта;
- просто и удобно истолковать, а, следовательно, и использовать данное описание;
- изобразить все элементы внешнего окружения системы, определить каким образом будет организовано функциональное взаимодействие между ними, а это в свою очередь позволит выявить ошибочную, дублируемую и поэтому лишнюю информацию;
- определить сопроводительную документацию, описывающую саму систему и порядок работы с ней;
- четко определить порядок действий при использовании системы после ввода ее в эксплуатацию;
- осуществить анализ функций уже существующих систем на наличие альтернативных путей решения задачи, что впоследствии позволит выработать список действий для улучшения системы и отобрать более эффективные решения.

В настоящее время существует две основных методологии для разработки структурно-функциональных моделей:

- Data Flow Diagrams (DFD);
- Icam DEFinition (IDEF0).

Каждая из перечисленных методологий использует графический язык описания процессов, поэтому любая модель, разработанная с их помощью, будет представлена в виде совокупности иерархически упорядоченных и взаимосвязанных диаграмм, каждая из которых является единицей описания системы и располагается на отдельном листе.

Выбор функциональной модели для создания ЭОР был сделан в пользу графической нотации IDEF0, так как она универсальная и позволит разработать наиболее полную и понятную модель ЭОР.

Графическая нотация IDEF0 предоставляет набор инструментов для создания диаграмм, каждый из которых имеют свое название и изображение (см. Рис.5)..

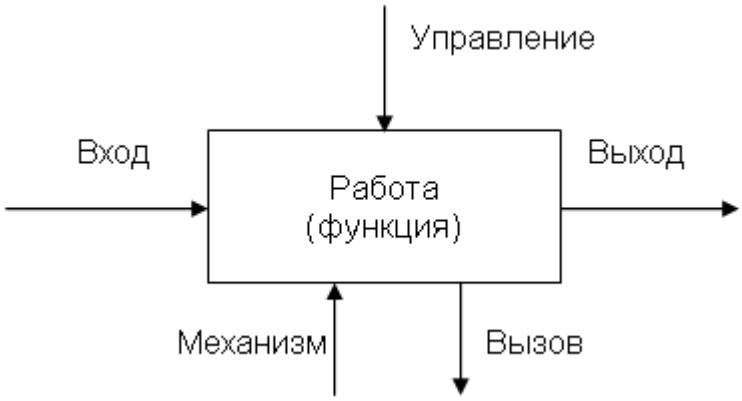


Рисунок 5. Элементы графической нотации IDEF0

Прямоугольник представляет собой ***работу (процесс, деятельность, функцию или задачу)***, которая имеет фиксированную цель и приводит к некоторому конечному результату. Имя работы должно выражать действие (например, «Изготовление детали», «Расчет допускаемых скоростей», «Формирование ведомости ЦДЛ № 3»).

Взаимодействие работ между собой и внешним миром описывается в виде стрелок. В IDEF0 различают ***5 видов стрелок***:

- ***вход*** (англ. input) – материал или информация, которые используются и преобразуются работой для получения результата (выхода). Вход отвечает на вопрос «Что подлежит обработке?». В качестве входа может быть как материальный объект (сырье, деталь, экзаменационный билет), так и не имеющий четких физических контуров (запрос к БД, вопрос преподавателя). Допускается, что работа может не иметь ни одной стрелки входа. Стрелки входа всегда рисуются входящими в левую грань работы;
- ***управление*** (англ. control) – управляющие, регламентирующие и нормативные данные, которыми руководствуется работа. Управление отвечает на вопрос «В соответствии с чем выполняется работа?». Управление влияет на работу, но не преобразуется ей, т.е. выступает в качестве ограничения. В качестве управления могут быть правила, стандарты, нормативы, расценки, устные указания. Стрелки управления рисуются входящими в верхнюю грань работы. Если при построении диаграммы возникает вопрос, как правильно нарисовать стрелку сверху или слева, то рекомендуется ее рисовать как вход (стрелка слева);
- ***выход*** (англ. output) – материал или информация, которые представляют результат выполнения работы. Выход отвечает на вопрос «Что является результатом работы?». В качестве выхода может быть как материальный объект (деталь, автомобиль, платежные документы, ведомость), так и нематериальный (выборка данных из БД, ответ на вопрос, устное указание). Стрелки выхода рисуются исходящими из правой грани работы;
- ***механизм*** (англ. mechanism) – ресурсы, которые выполняют работу. Механизм отвечает на вопрос «Кто выполняет работу или

посредством чего?». В качестве механизма могут быть персонал предприятия, студент, станок, оборудование, программа. Стрелки механизма рисуются входящими в нижнюю грань работы;

- ***вызов*** (англ. call) – стрелка указывает, что некоторая часть работы выполняется за пределами рассматриваемого блока. Стрелки выхода рисуются исходящими из нижней грани работы.

С помощью набора данных функциональных компонент, каждая из которых имеет стандартное графическое изображение и осуществляется описание системы в виде диаграмм, которые связаны между собой потоками данных.

Таким образом, с помощью элементов графической нотации IDEF0 была создана функциональная модель (0) на основе которой был разработан ЭОР «Поиск путей обхода графа».

В результате, на основе построенной функциональной модели был разработан ЭОР «Поиск путей обхода графов».

Создание ЭОР происходило на языке объектно-ориентированного программирования C#, посредством функциональных возможностей среды разработки Visual Studio. Подробный листинг ресурса представлен в приложении 2 (**Ошибка! Источник ссылки не найден.**).

Разработанный ЭОР представляет собой объект Windows Form, который разделен на три области: область панели инструментов, область визуализации графа и область для вывода результатов. При запуске ЭОР на экране отображается главная область для построения графа (см. Рис.6).

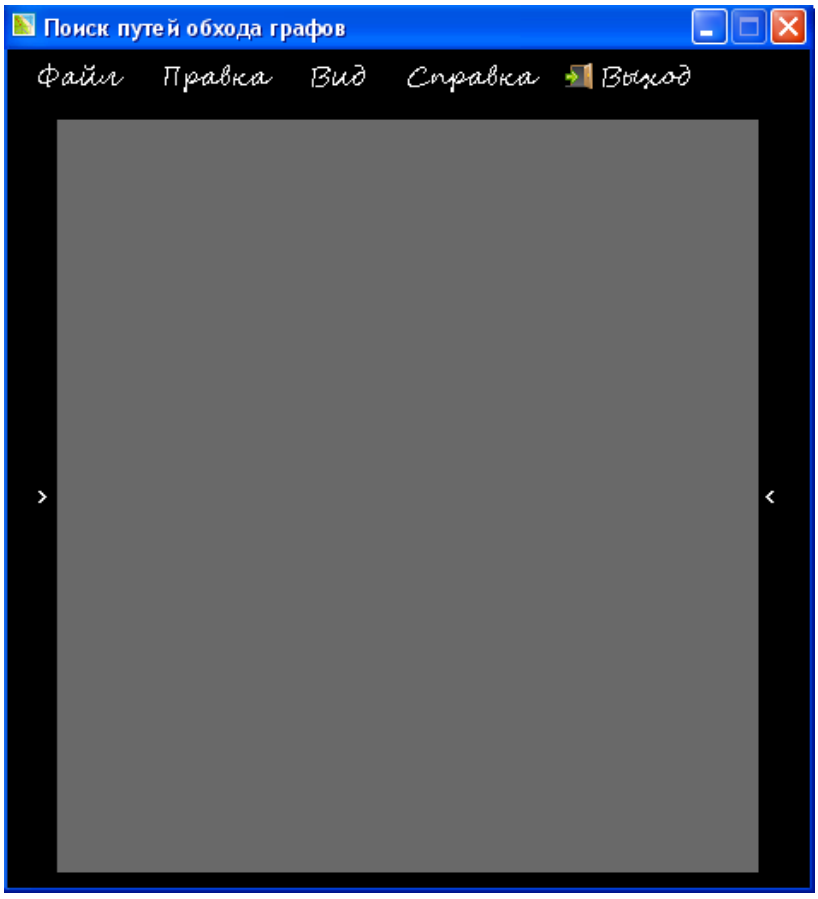


Рисунок 6. Внешний вид ЭОР « Поиск путей обхода графов" после запуска

Основная функция данного ЭОР состоит в том, чтобы предоставить студенту инструменты, посредством которых он сможет визуально построить граф в специально отведенной для этого области. Таким образом, чтобы изобразить граф, следует воспользоваться панелью инструментов, которая находится слева от главного поля и открывается по нажатию кнопки со стрелкой (см. Рис.7).

Рисунок 7. Внешний вид панели инструментов ЭОР

Первый инструмент с изображением стрелки означает, что можно выбрать ранее нарисованную вершину и переместить ее. При выборе любой вершины она будет очерчена красным цветом, в случае, когда зажата данная кнопка. Кроме того в специальном поле, которое отображается справа при нажатии на соответствующую кнопку со стрелкой, можно увидеть степень выбранной вершины. Но данная функция доступна только в том случае, если в области изображения графа расположено как минимум две вершины и одно ребро.

Вторая кнопка с точкой используется для расположения вершины в области визуализации. То есть для изображения вершины, следует нажать на данную кнопку, после чего кликнуть мышкой в любом месте, где должна быть изображена вершина. Вершины нумеруются автоматически.

Кнопка с карандашом позволяет соединить вершины ребрами. Перед построением ребер следует выбрать, каким именно будет изображаемый граф, то есть при нажатии правой кнопок мыши на кнопку с карандашом появится выпадающее меню, в котором предоставляется выбор таких полей как «Орграф» или «Взвешенный граф» (см. Рис.8). Так же можно отметить галочками оба этих пункта одновременно. В том случае, если ни один из пунктов останется не отмеченным, то все построенные ребра не будут иметь ни направления, ни веса. Ребро изображается с помощью щелчка на первой вершине (она станет выделена другим цветом), а затем на второй.

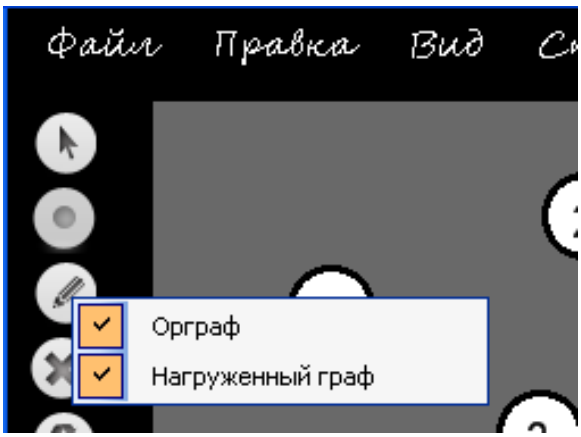


Рисунок 8. Внешний вид выпадающего меню при выборе вида графа

Две следующие кнопки отвечают за удаление элемента графа либо всего графа полностью. Для удаления элемента следует зажать соответствующую кнопку и щелкнуть на элементе, который требуется удалить, при этом нумерация вершин и ребер обновится. При удалении всего графа, сначала появится окно, требующее подтверждения удаления (см. Рис.9) и в случае выбора положительного ответа пользователем, граф удаляется.

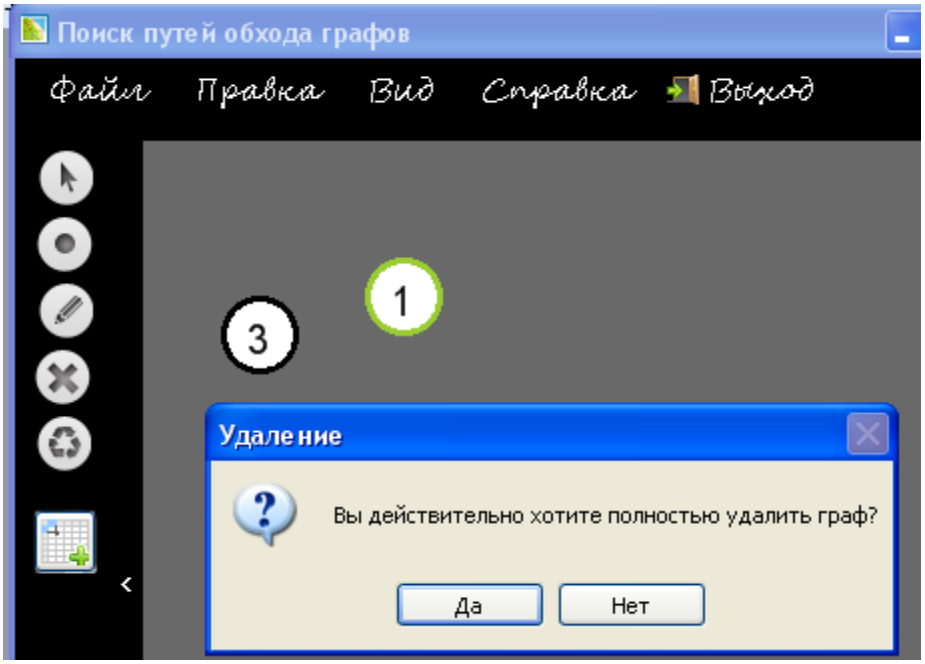


Рисунок 9. Диалоговое сообщение при удалении графа

Таким образом, выбирая инструменты на панели, нужно расположить вершины на поле в центре формы, и соединить эти вершины ребрами, то есть построить граф (см. Рис.10).

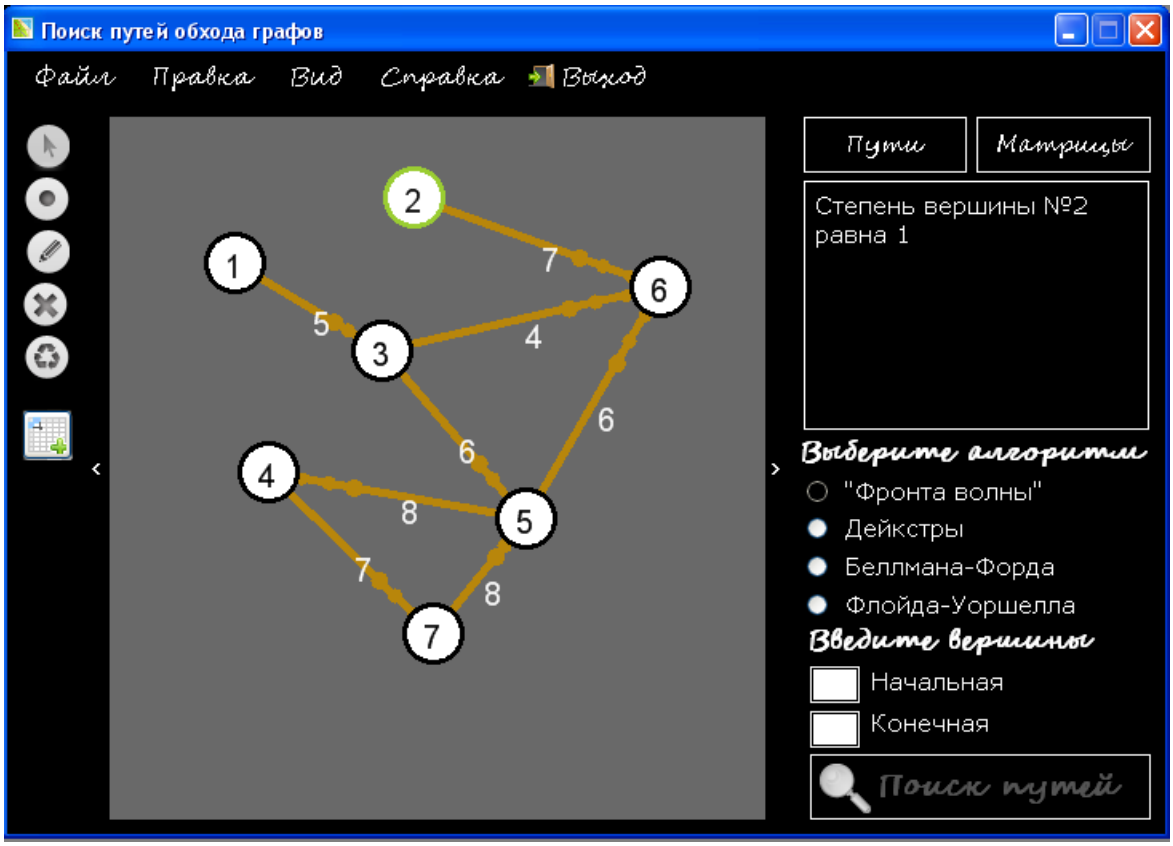


Рисунок 10. Внешний вид приложения после построения графа

Еще одна кнопка с изображением таблицы на панели инструментов отвечает за построение матриц смежности и инцидентности. Чтобы выбрать нужную матрицу следует вызвать всплывающее меню щелчком правой кнопки мыши на кнопке матриц. В результате в специальной таблице построится выбранная матрица (см. Рис.11 и Рис.12).

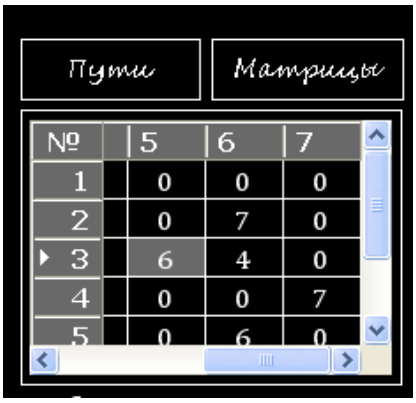


Рисунок 11. Внешний вид построенной матрицы смежности

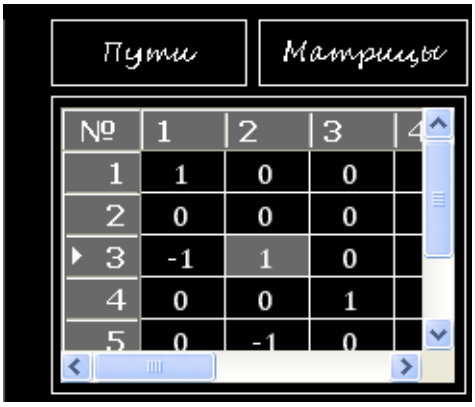


Рисунок 12. Внешний вид построенной матрицы инцидентности

После построения графа, для того чтобы найти в нем кратчайшие пути следует выбрать алгоритм по которому будет производиться поиск (см. Рис.13). В списке будут доступны для выбора только те алгоритмы, которые применимы для изображенного графа. Причем каждый из алгоритмов решает определенную задачу о нахождении кратчайшего пути, поэтому при затруднении с выбором алгоритма следует изучить теорию, касающуюся основ теории графов и алгоритмов, которую можно вывести на экран при помощи выбора пункта меню Справка. От задачи решаемой алгоритмом, зависит, нужно ли вводить информацию в текстовые поля Начальная вершина и Конечная вершина, которые могут быть как доступны для ввода информации, так и нет.

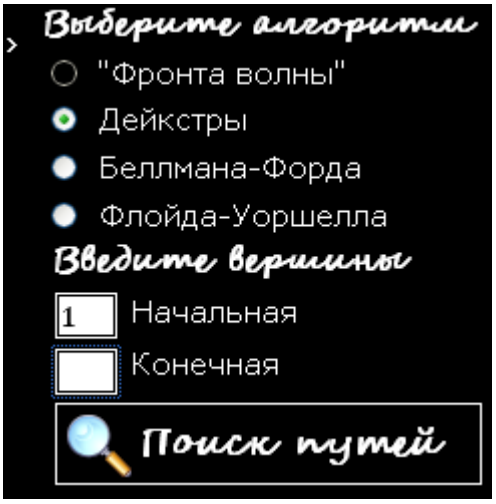


Рисунок 13. Внешний вид области для выбора алгоритма и ввода параметров.

После выбора алгоритма и ввода необходимой информации о вершинах, станет активной кнопка Поиск пути, по нажатию на которую, происходит вывод найденных путей или пути, в зависимости от выбранного алгоритма, в

специальной области расположенной там же где была построена матрица смежности или инцидентности (см. Рис.14). При этом можно осуществить переход к построенной матрице и обратно к результатам, для этого над полями для вывода расположены две кнопки, которые в данном случае используются как вкладки.

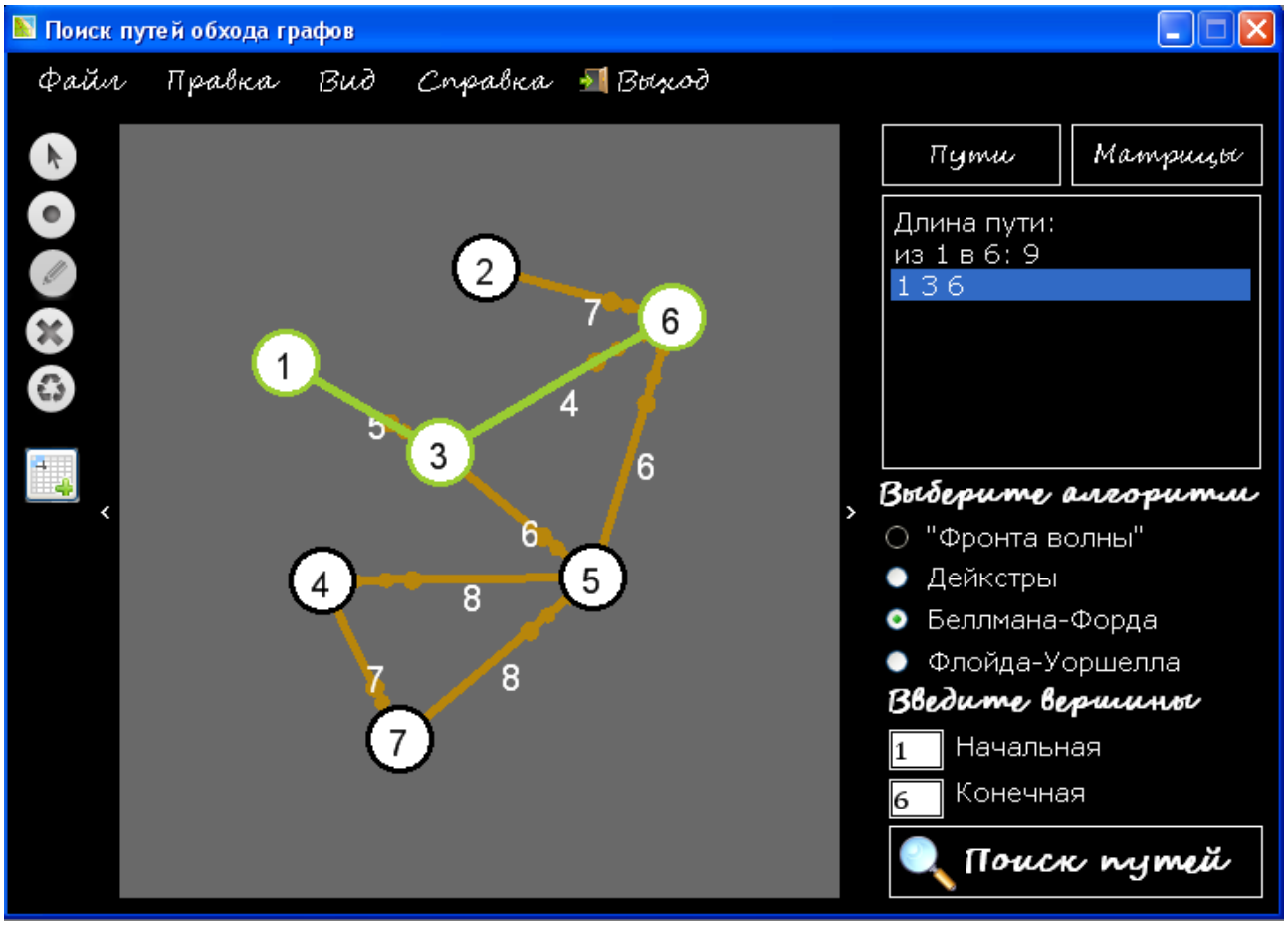


Рисунок 14. Внешний вид электронного образовательного ресурса по поиску путей обхода графов после построения графа и вывода списка путей

Список путей полученный по выбранному алгоритму можно сохранить в файл формата *.txt или *.doc, который впоследствии можно открыть с помощью любого текстового редактора позволяющего работать с файлами данного формата. Для этого следует в пункте главного меню Файл, выбрать подпункт Сохранить и далее Список путей. Таким образом, список путей будет сохранен в файл нужного формата в нужную папку.

Таким же образом можно в любой момент сохранить изображение построенного графа в графический файл, который можно будет открыть

стандартной (или любой другой) программой для просмотра изображений. Для этого следует в пункте главного меню Файл, выбрать подпункт Сохранить и далее Граф. Далее в диалоговом окне выбрать формат файла, дать название и выбрать путь сохранения. Таким образом, изображение графа будет сохранено в файл в нужную папку (см. Рис.15).

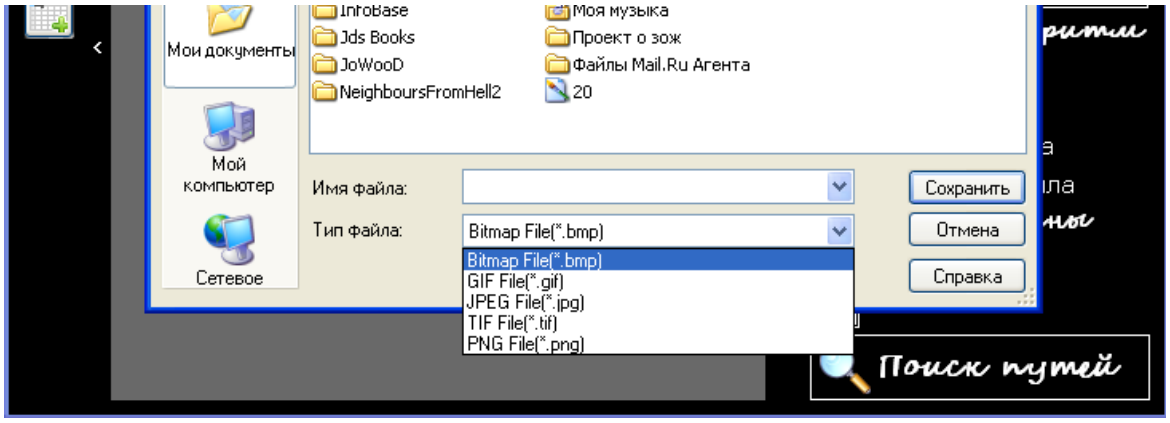


Рисунок 15. Внешний вид диалогового окна при сохранении изображения графа

Для вызова справки и просмотра материалов по теории графов следует воспользоваться пунктом меню Справка (см. Рис.16).

В меню Правка можно снять выделение с вершины или всего графа, а меню Вид позволяет скрывать или отображать боковые панели.

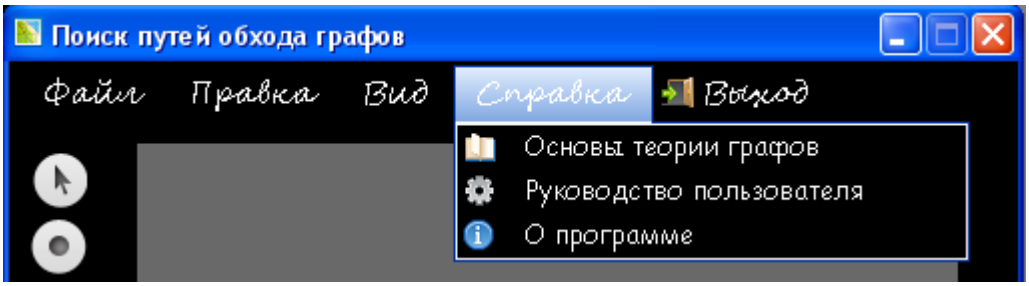


Рисунок 16. Вид пункта Справка главного меню ЭОР

Апробация разработанного ЭОР «Поиск путей обхода графов» была осуществлена в процессе освоения учебной дисциплины «Объектно-ориентированное программирование в решении математических задач» студентами группы БИ-31, а так же студентами групп Б-41 и БИ-41, которые изучали данную дисциплину ранее.

При апробации был использован метод экспертных оценок. Экспертами выступили студенты, которые с помощью предоставленной им анкеты,

созданной с помощью онлайн-сервиса Google Forms, оценили такие характеристики ЭОР как эффективность, удобство интерфейса и качество полученных результатов. Таким образом, в результате апробации с помощью анкеты было опрошено 24 студента, из которых 45,8% учащиеся 3 курса и 54,2% учащиеся 4 курса.

По представленным диаграммам (см. Рис.17) можно сделать вывод о том, что практически все студенты склоняются к ответу, что данный ЭОР был или мог бы быть полезен им при решении задач по данной дисциплине. Кроме того большинство из характеристик представленного ЭОР были оценены на высшем уровне (см. Рис.18, 19). Так же из пяти студентов, которым понадобилось воспользоваться руководством пользователя или материалами по теории графов, четыре нашли нужную информацию в полном объеме и только один частично (см. Рис.20).

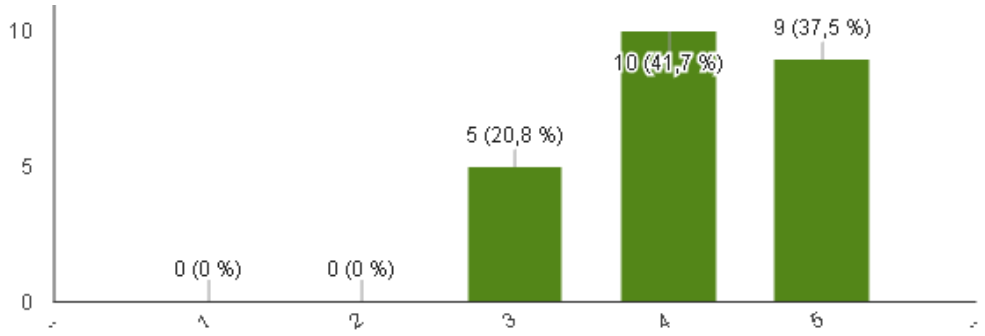


Рисунок 17.Диаграмма полученных ответов на вопрос о полезности ЭОР «Поиск путей обхода графа»

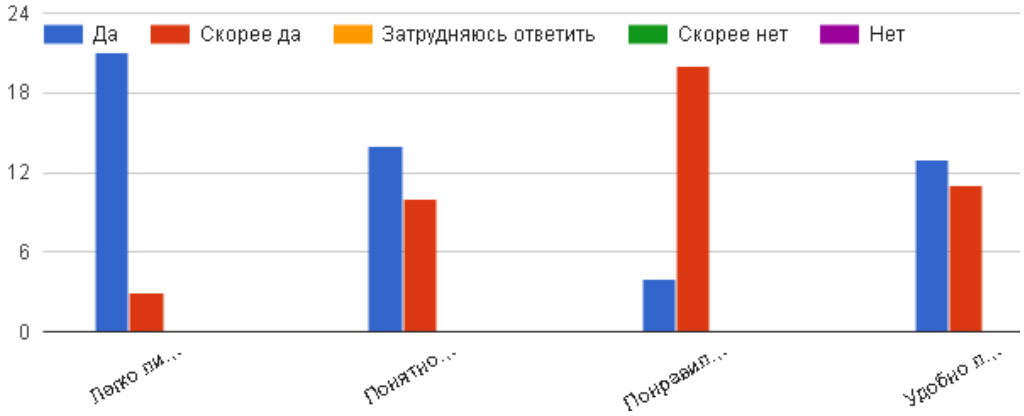


Рисунок 18. Диаграмма полученных ответов при оценке характеристик ЭОР «Поиск путей обхода графа»

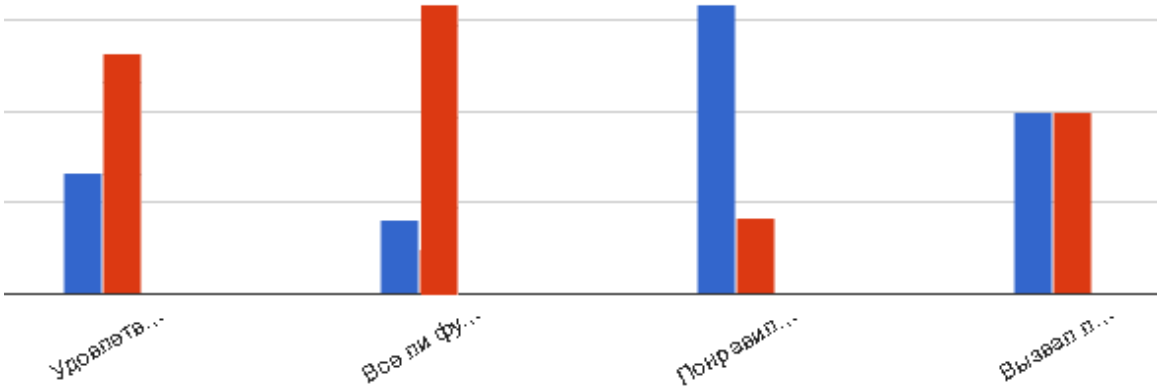


Рисунок 19. Диаграмма полученных ответов при оценке характеристик ЭОР «Поиск путей обхода графа»

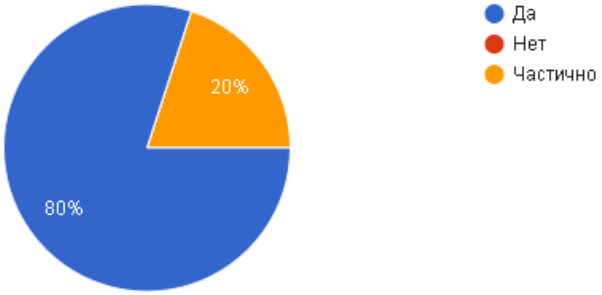


Рисунок 20 .Диаграмма полученных ответов на вопрос о поиске нужной информации в представленных материалах по основам теории графов.

Основываясь на результатах данного анкетирования можно говорить о том что, по мнению студентов данный ЭОР удобен, прост и понятен в использовании, выбранная цветовая схема не отвлекает от работы и не создает дополнительную нагрузку на глаза. Следовательно, при изучении дисциплины «Объектно-ориентированное программирование в решении математических задач» ЭОР «Поиск пути» будет способствовать повышению интереса к дисциплине, облегчит освоение трудного материала студентами при помощи визуальных средств, а так же станет удобным инструментом при решении математических задач.

Заключение

Современный темп жизни с каждым днем требует от современного общества нахождения все большего количества новых путей для опережающей разработки инновационных технологий производства. Для решения данной задачи не обойтись без достаточного количества инженерных кадров высокой квалификации. Таким образом, в настоящее время существует необходимость развития у студентов, осваивающих точные и естественные науки, инженерного мышления, результатом которого являются инновационные идеи, внедрение которых обеспечивает повышение эффективности производства.

В связи с этим можно говорить о том, что именно ЭОР по визуализации путей обхода графов будет способствовать повышению интереса обучающихся к изучению предметов по инженерным специальностям и улучшению качества подготовки специалистов в системе высшего образования данной области.

Для выполнения выпускной квалификационной работы была поставлена цель: разработать электронный образовательный ресурс для визуализации алгоритмов, основанных на графах, для решения инженерных и математических задач.

Анализ научно-методической литературы в области методов и средств развития инженерного мышления показал, что именно электронный образовательный ресурс, содержание которого отражает теоретическое обоснование методов обхода графов, а форма представления информации обеспечивает визуализацию этих методов, может применяться в качестве средства формирования и развития инженерного мышления. Кроме того были определены требования, предъявляемые к ЭОР и методы оценки их педагогико-эргономических качеств.

Выбор технологии реализации ЭОР был сделан в пользу языка объектно-ориентированного программирования C# и среды разработки Visual Studio 2010, так они обладает рядом существенных преимуществ, которые являются

важным аспектом для разработки ЭОР по визуализации поиска пути обхода графов.

В результате выполнения ВКР был разработан ЭОР направленный на овладение студентами на теоретическом и практическом уровнях методами решения графовых задач, с помощью средств компьютерной визуализации. Так же была разработана программная документация и составлен комплекс инженерных задач.

В ходе реализации ЭОР и для достижения поставленной цели были выполнены все задачи, сформулированные в начале выполнения работы

Литература

1. C# для вундеркиндов. Часть 4. Программирование в .NET Framework // msdn.microsoft.com URL: https://msdn.microsoft.com/ru-ru/library/bb297402(v=vs.80).aspx (дата обращения: 07.03.2016).
2. Edward V. Krick Введение в инженерное дело. Пер. с англ. - М.: «Энергия», 1970. - 176 с.
3. Алгоритм Дейкстры // algolist.manual.ru URL: http://algolist.manual.ru/maths/graphs/shortpath/dijkstra.php (дата обращения: 20.03.2016).
4. Алгоритм Дейкстры. // e-maxx.ru URL: http://e-maxx.ru/algo/dijkstra (дата обращения: 22.02.2016).
5. Алгоритм Флойда // algolist.manual.ru URL: http://algolist.manual.ru/maths/graphs/shortpath/floyd.php (дата обращения: 27.03.2016).
6. Алгоритм Флойда-Уоршелла // e-maxx.ru URL: http://e-maxx.ru/algo/floyd_warshall_algorithm (дата обращения: 28.02.2016).
7. Алгоритм Форда-Беллмана // algolist.manual.ru URL: http://algolist.manual.ru/maths/graphs/shortpath/ford.php (дата обращения: 23.03.2016).
8. Алгоритм Форда-Беллмана // e-maxx.ru URL: http://e-maxx.ru/algo/ford_bellman (дата обращения: 25.02.2016).
9. Алгоритмы на графах. Рабочая программа дисциплины / Глебов А.Н. - Новосибирск: ФИТ НГУ, 2003. – 5с.
10. Артемова Д.Т. Программа предпрофильной подготовки "Полезная математика. Приложение теории графов" // – М.: 2004.
11. В.А.Евстигнеев, под редакцией А.П. Ершова. Применение теории графов в программировании. – М.: Главная редакция физико-математической литературы, 1985. – 350 с.

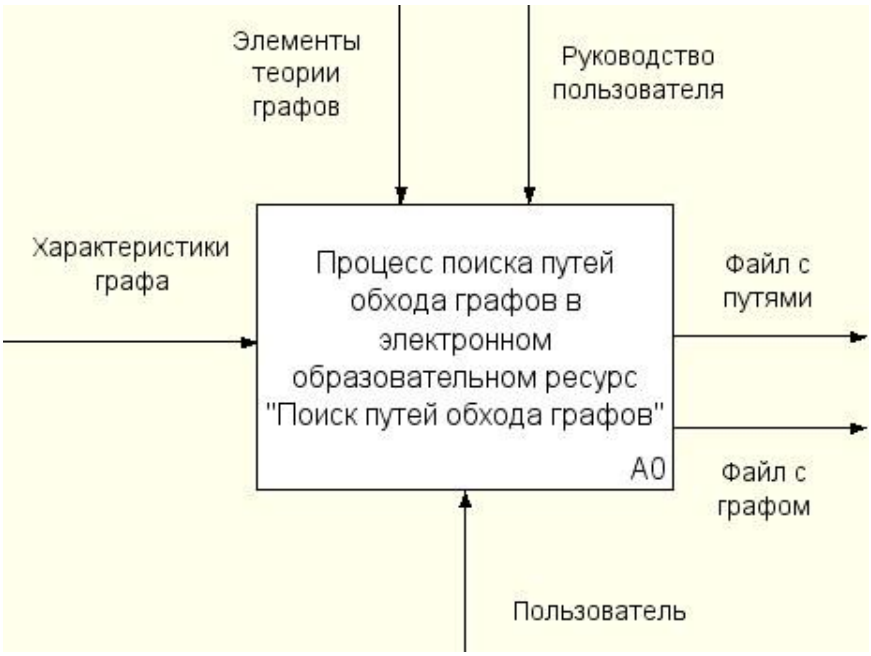
12. Введение в язык C# и .NET Framework // msdn.microsoft.com URL: https://msdn.microsoft.com/ru-ru/library/z1zx9t92.aspx (дата обращения: 14.03.2016).
13. Волновой алгоритм // algolist.manual.ru URL: http://algolist.manual.ru/maths/graphs/shortpath/wave.php (дата обращения: 03.03.2016).
14. Газейкина А.И. Обучение будущего учителя информатики конструированию учебных заданий, направленных на формирование метапредметных результатов обучения // Педагогическое образование в России. - 2014. - №18. - С. 9.
15. Занфирова Л.В. Формирование технического мышления в процессе подготовки студентов агроинженерных вузов: автореф. дис. канд. пед. наук: 13.00.08. - М., 2008. – 24 с.
16. Зыков А.А. Основы теории графов. - М.: «Вузовская книга», 2004. - 664 с.
17. Использование среды разработки Visual C# // msdn.microsoft.com URL: https://msdn.microsoft.com/ru-ru/library/ms173063.aspx (дата обращения: 15.03.2016).
18. Лекция 9. Кратчайшие пути в графе // studfiles.ru URL: http://www.studfiles.ru/preview/3622033/ (дата обращения: 10.02.2016).
19. Национальный стандарт Российской Федерации " Информационно-коммуникационные технологии в образовании. Термины и определения" от 27 декабря 2006 № 52653-2006 // Стандартинформ. - 2007 г.
20. Национальный стандарт Российской Федерации "Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению" от 28 декабря 1993 № ИСО/МЭК 9126-93 // Издательство стандартов. - 1994 г.
21. Обоснование требований к мультимедийным образовательным ресурсам // scienceforum.ru: Электронный научный журнал «Международный

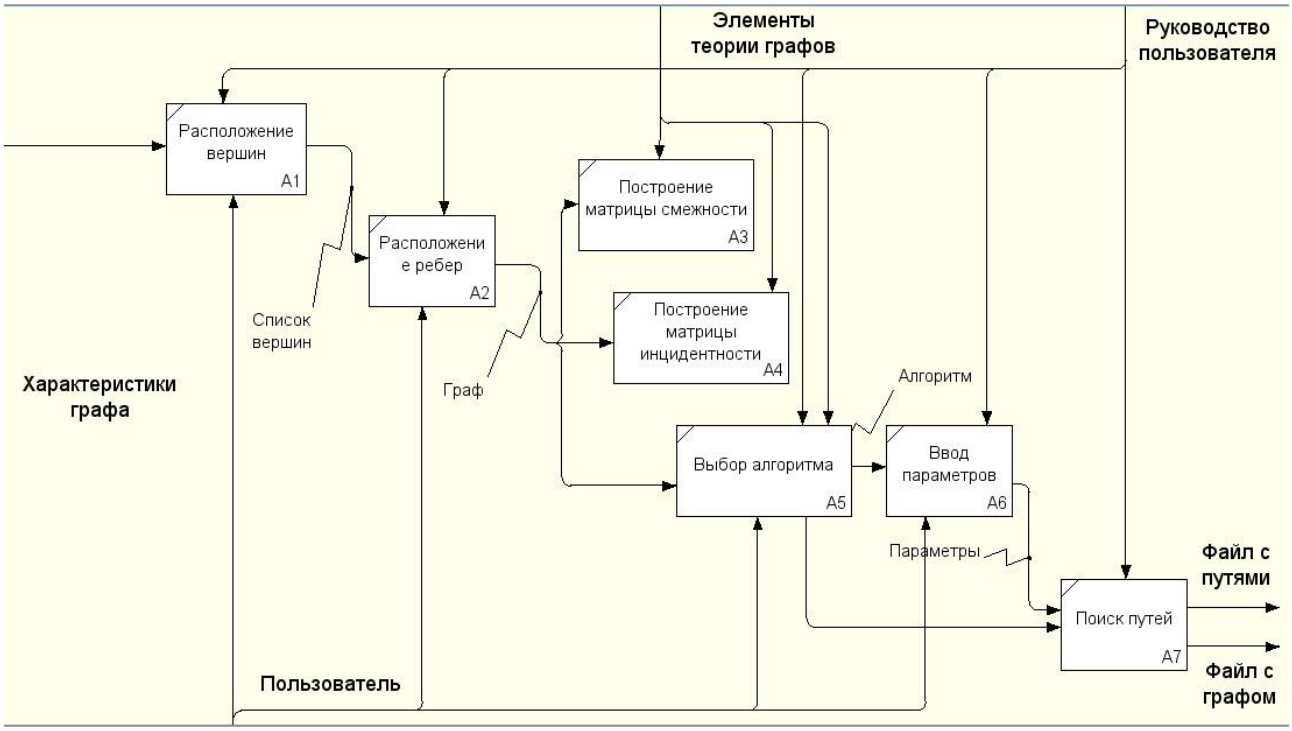
студенческий научный вестник» URL: http://www.scienceforum.ru/2014/596/3354 (дата обращения: 05.03.2016).

22. Понятие функциональной модели предметной области базы данных // intuit.ru: Национальный открытый университет ИНТУИТ URL: http://www.intuit.ru/studies/courses/1095/191/lecture/4969?page=4 (дата обращения: 18.03.2016).
23. Сазонова З.С., Чечеткина Н.В. Развитие инженерного мышления – основа повышения качества образования: Учебное пособие / МАДИ (ГТУ). –М.: 2007. –195 с.
24. Стась А.Н., Прусских О. Н. Формирование алгоритмического мышления в процессе обучения теории графов // Вестник Томского государственного педагогического университета. – 2012. – №2. – 166-169с.
25. Татрова А.С. Психология. - М.: Академия Естествознания, 2010. – 284 с.
26. Троелсен. Э. Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд. : Пер. с англ — М : ООО "И.Д. Вильяме", 2011. — 1392 с.
27. Указ Губернатора Свердловской области "О комплексной программе "Уральская инженерная школа"" от 6 октября 2014 г. № 453-УГ // Официальный интернет-портал правовой информации Свердловской области. http://www.pravo.gov66.ru. 2014 г. – 24 с.
28. Умелов А. Н., Берштейн Л. С., Курейчик В. М. Применение графов для проектирования дискретных устройств: эл. библ. МГУ // – М.: 2001.
29. Формирование инженерного мышления в процессе обучения: материалы междунар. науч-практ. конф.,7-8 апреля 2015г., Екатеринбург: / Урал. гос.пед. ун-т; отв. ред. Т.Н. Шамало. – Екатеринбург: [б.и.], 2015.–284с.
30. Электронный образовательный ресурс // megabook.ru: Мегаэнциклопедия Кирилла и Мефодия URL: http://megabook.ru/article (дата обращения: 28.03.2016).

Приложения

Приложение 1.
Функциональная модель

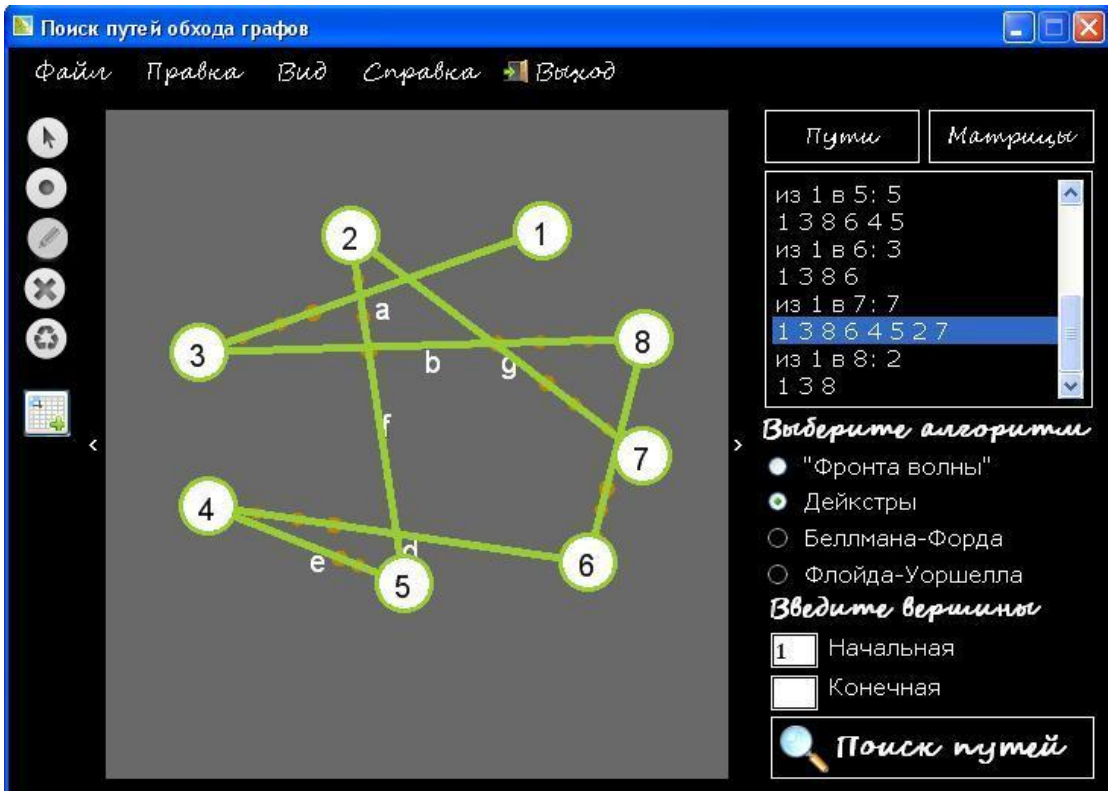




Приложение 2.
Комплекс задач и их решений

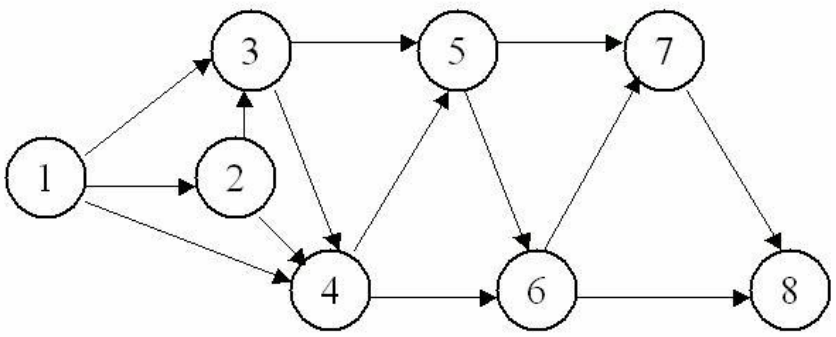
Задача 1. На пришкольном участке растут 8 деревьев: яблоня, тополь, береза, рябина, дуб, клен, лиственница и сосна. Рябина выше лиственницы, яблоня выше клена, дуб ниже березы, но выше сосны, сосна выше рябины, береза ниже тополя, а лиственница выше яблони. Расположите деревья от самого высокого к самому низкому.

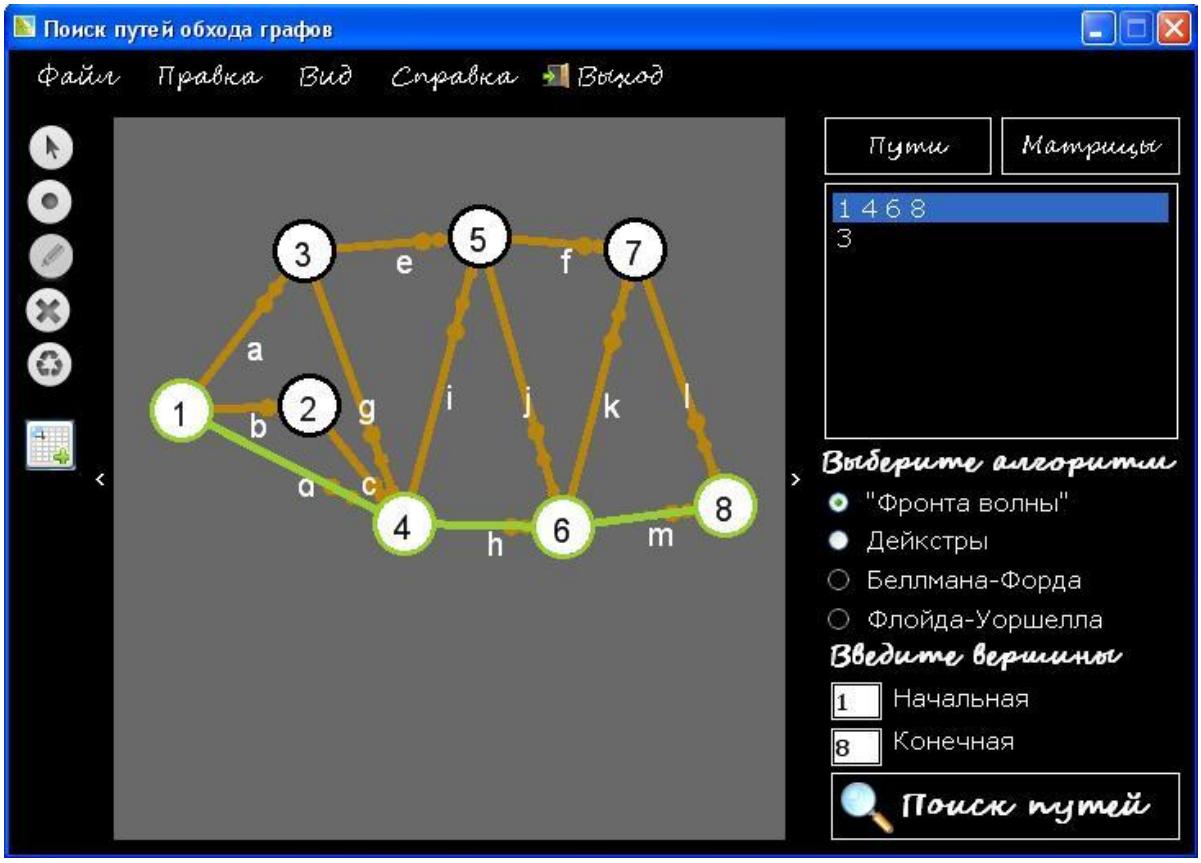
Решение



Задача 2.

Найти кратчайший путь из вершины 1 в вершину 8.

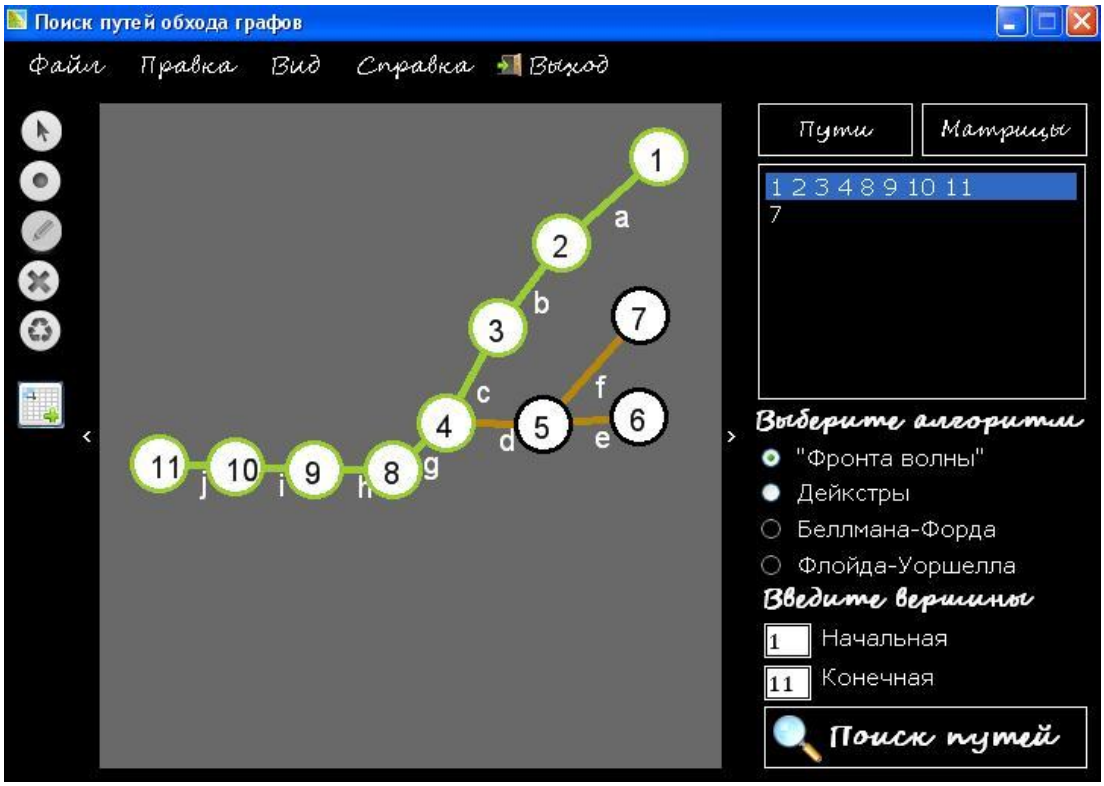




Задача 3. Дана схема метро города Екатеринбурга, которая включает линии, которые планируется построить в будущем.

Требуется определить, сколько станций потребуется проехать, чтобы попасть со станции Машиностроителей до станции Верх-Исетская.

Решение



Задача 4.

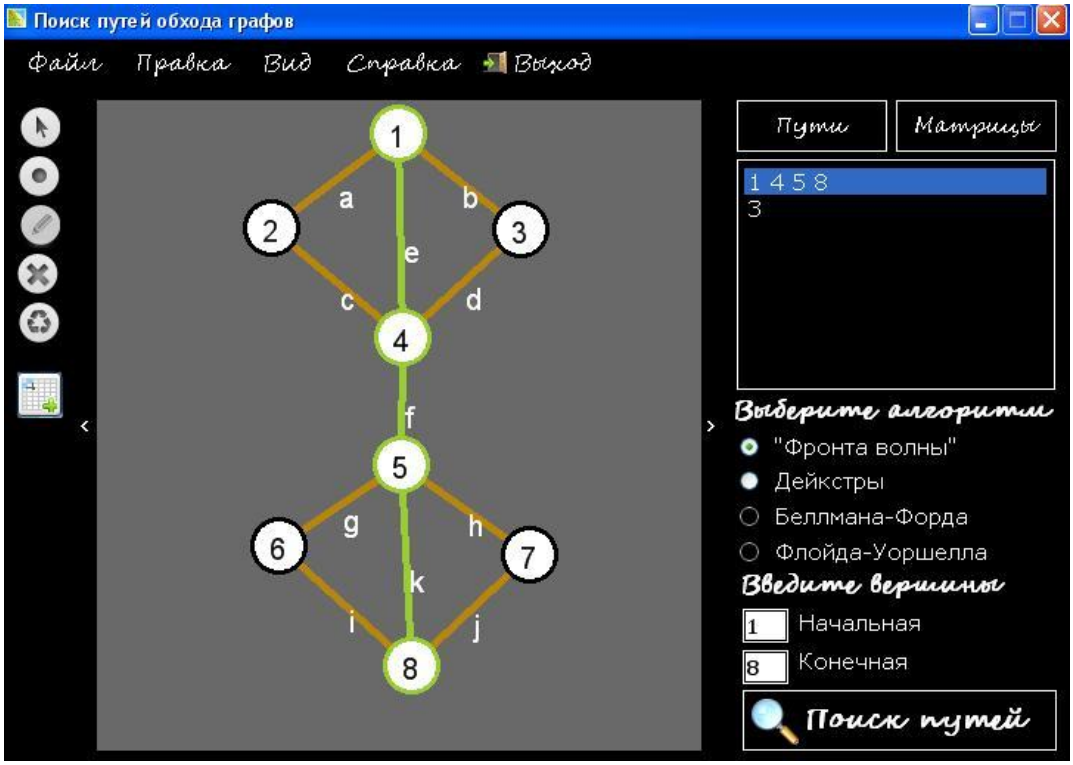
Требуется произвести действие: скопировать и вставить текст в документе открытом в текстовом редакторе Word. Для этого нужно выполнить некий алгоритм из нескольких действий:

- выделить текст, нажать на правую кнопку мыши и нажать Копировать;
- встать на место где нужно вставить текст, нажать на правую кнопку мыши и нажать вставить;
- выделить текст, нажать кнопку Копировать на панели инструментов Стандартная;
- встать на место где нужно вставить текст, нажать кнопку Вставить на панели инструментов Стандартная;
- выделить текст, зайти в меню Правка, выбрать пункт Копировать;

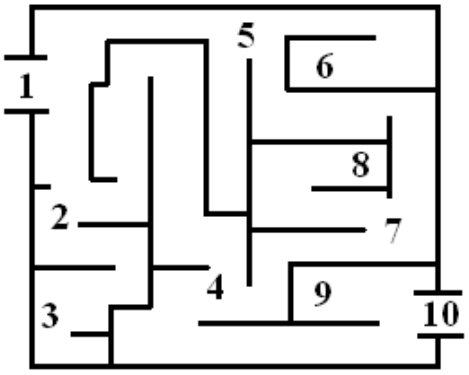
- встать на место где нужно вставить текст, зайти в меню Правка, выбрать пункт Вставить.

Требуется определить алгоритм копирования и вставки текста, который выполнится за минимальное количество действий.

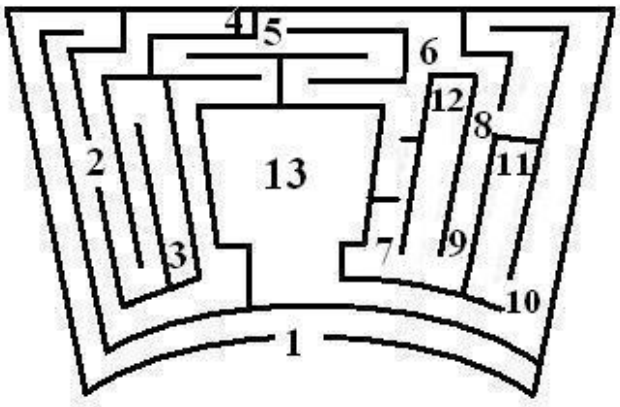
Решение



Задача 5.Рассмотрим задачу о поиске выхода из лабиринта, коридоры которого не обязательно находятся на одном уровне. Подобная ситуация возникает, например, при блуждании в пещерах или катакомбах. Коридоры лабиринта – это ребра графа, а перекрестки, тупики, входы и выходы – это вершины. Найти кратчайший выход из лабиринта.

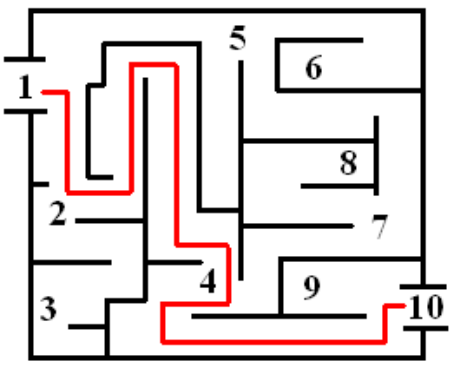


а)

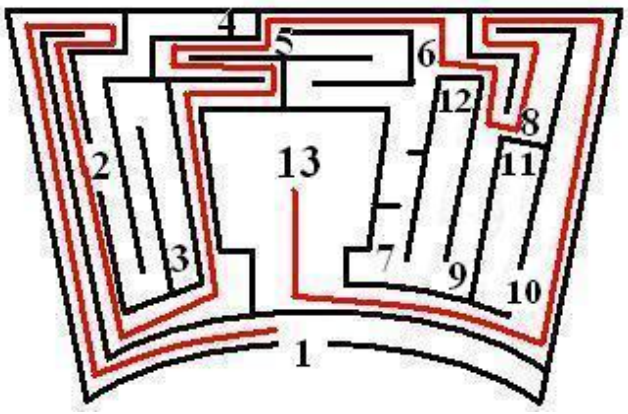


б)

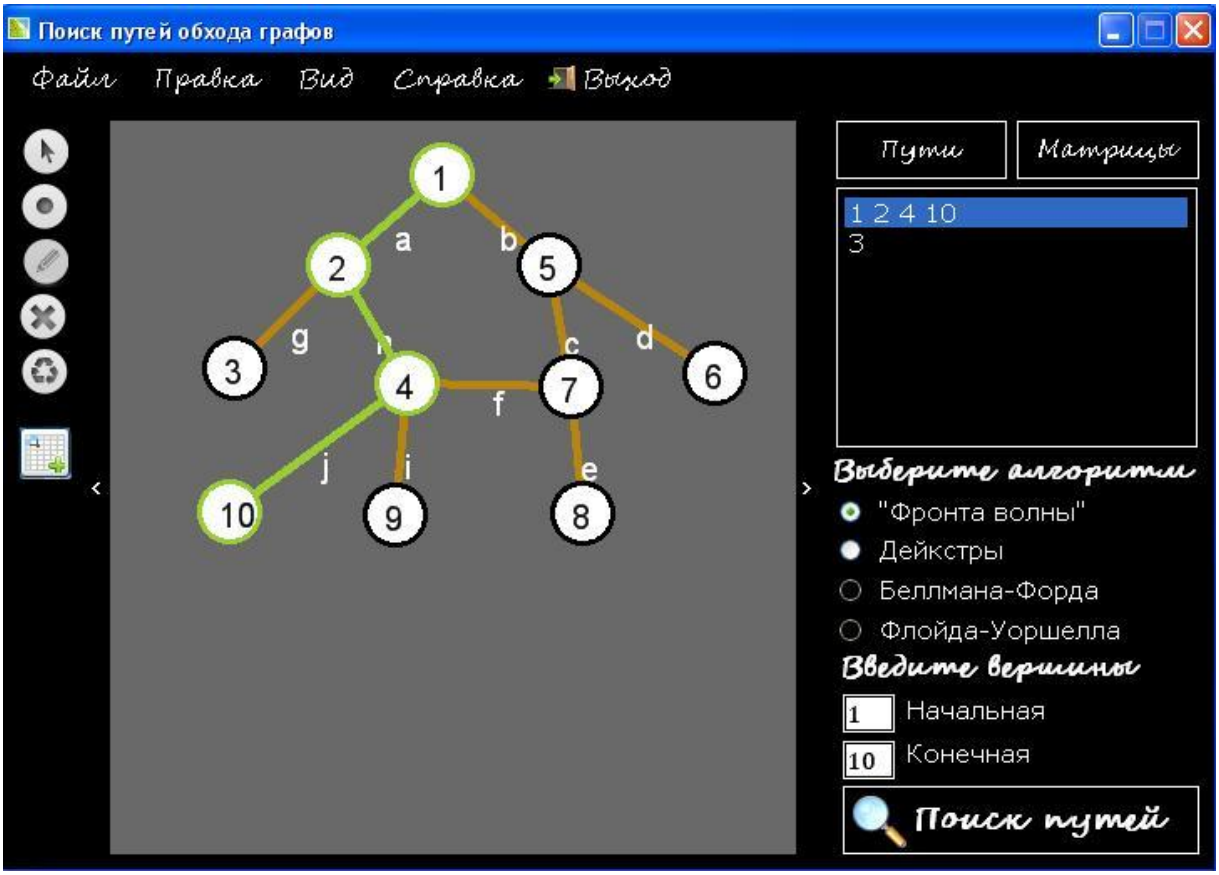
Решение

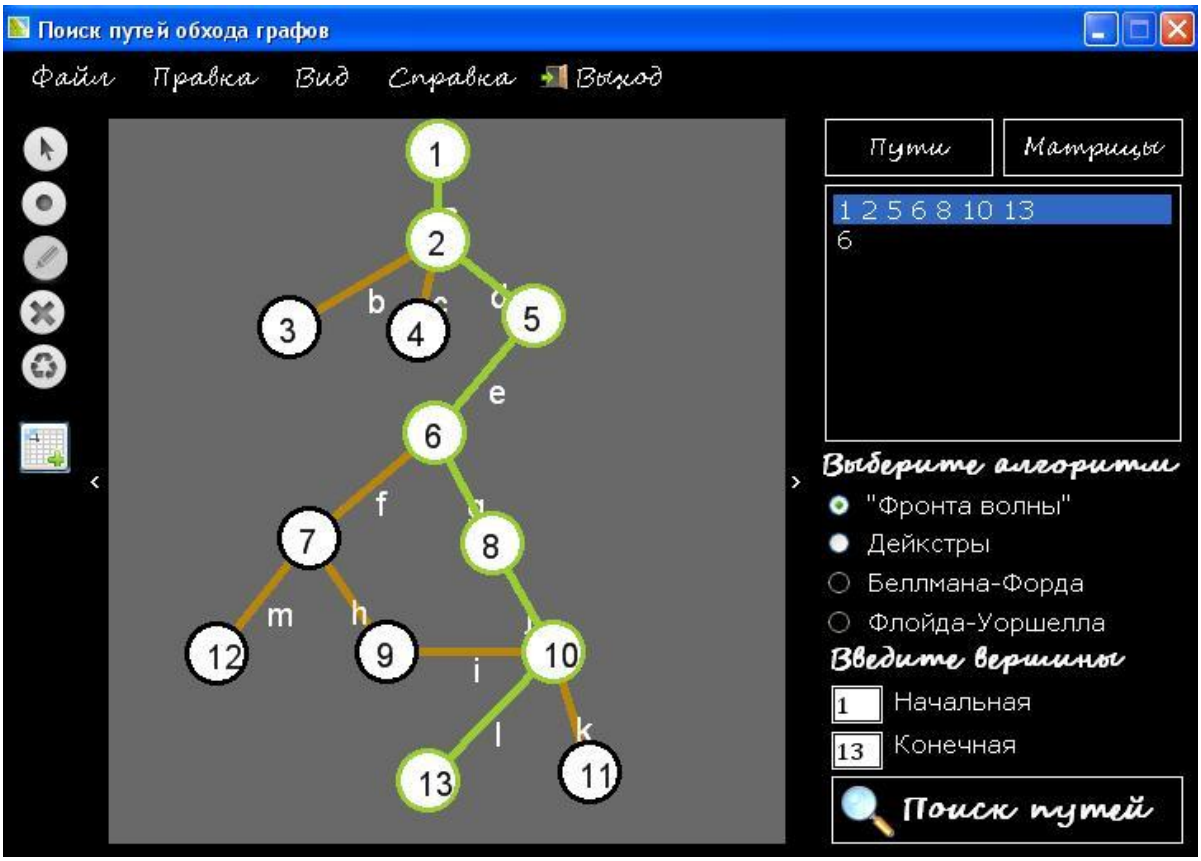

1
2
3
4
5
6
7
8
9
10

а)

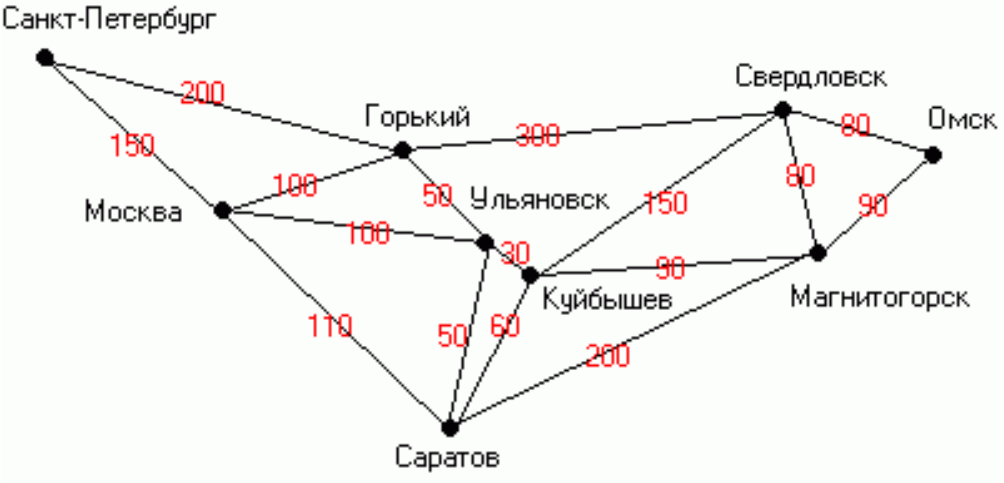

1
2
3
4
5
6
7
8
9
10
11
12
13

б)


Поиск путей обхода графов
Файл
Правка
Вид
Справка
Выход
Пути
Матрицы
1 2 4 10
3
Выберите алгоритм
"Фронта волны"
Дейкстры
Беллмана-Форда
Флойда-Уоршелла
Введите вершины
1
Начальная
10
Конечная
Поиск путей
a
b
c
d
e
f
g
h
i
j



Задача 6. Дана карта дорог между городами, где указана длина каждой дороги.



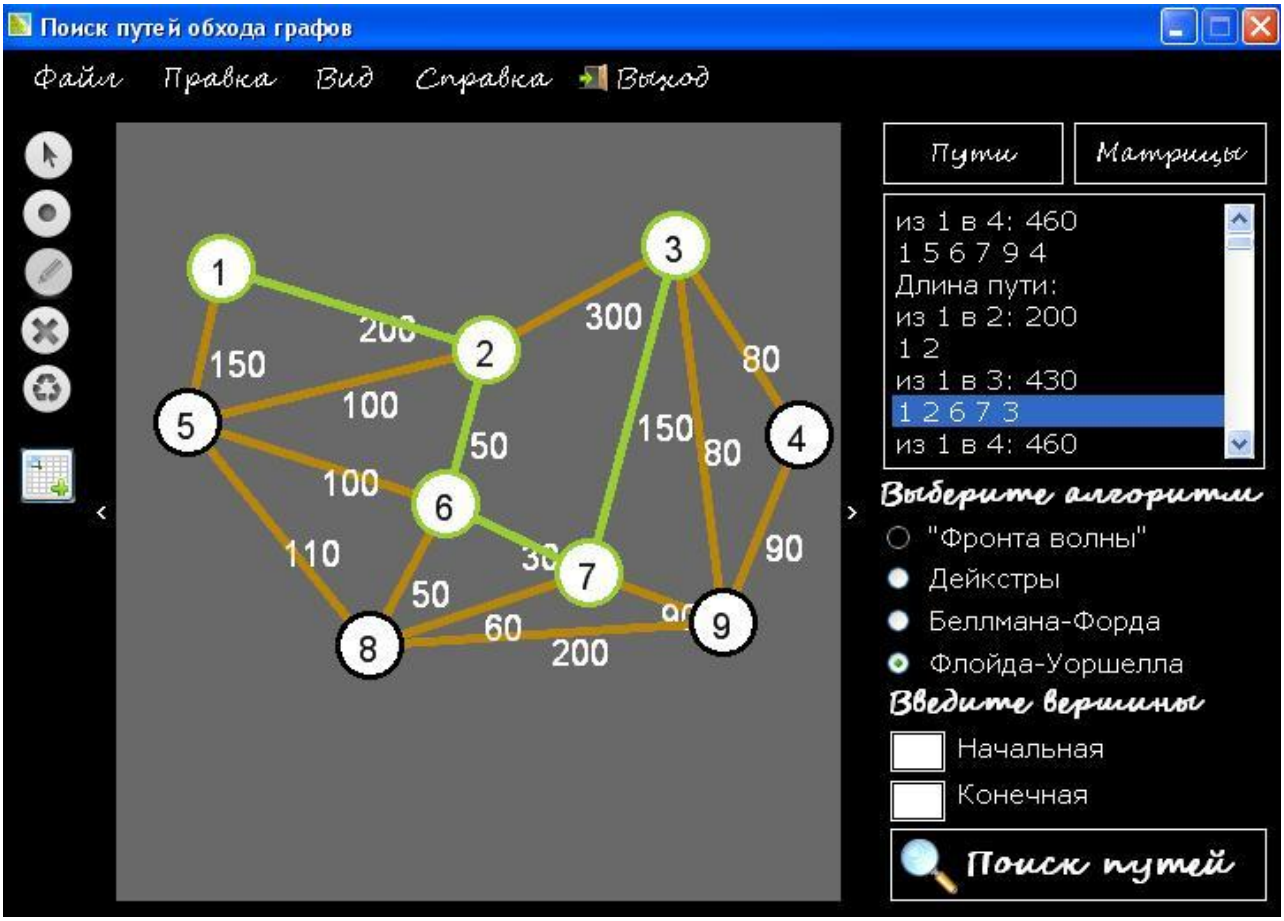
Найти:

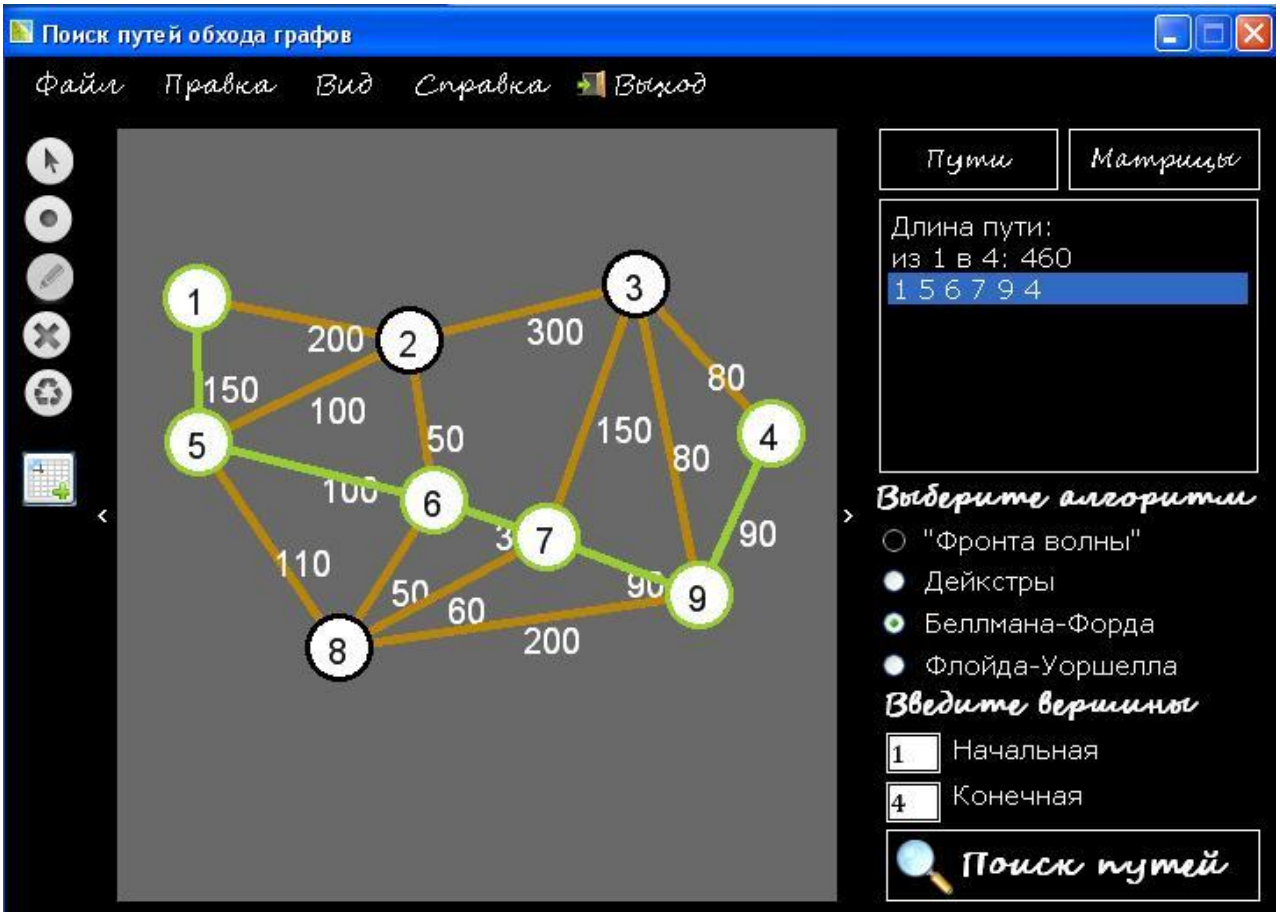
а) кратчайший путь из Санкт-Петербурга до Омска;

б) кратчайший путь из Санкт-Петербурга до Свердловска.

в) кратчайшие пути из каждого города до каждого.

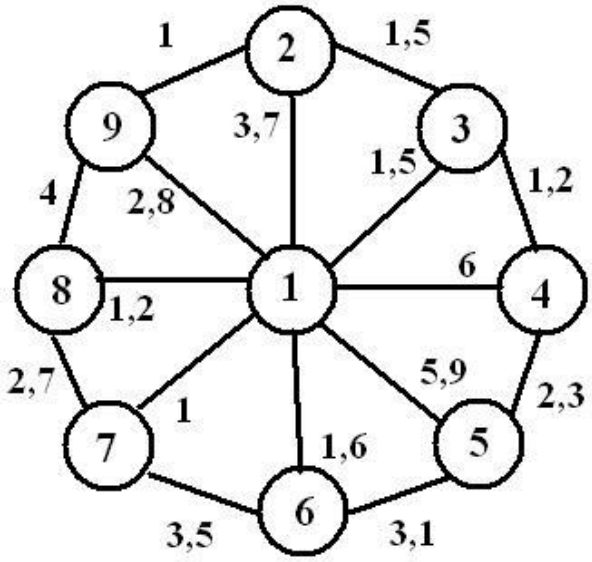
Решение


Поиск путей обхода графов
Файл
Правка
Вид
Справка
Выход
Пути
Матрицы
из 1 в 4: 460
1 5 6 7 9 4
Длина пути:
из 1 в 2: 200
1 2
из 1 в 3: 430
1 2 6 7 3
из 1 в 4: 460
Выберите алгоритм
"Фронта волны"
Дейкстры
Беллмана-Форда
Флойда-Уоршелла
Введите вершины
Начальная
Конечная
Поиск путей
1
2
3
4
5
6
7
8
9
200
300
150
80
100
50
150
80
100
110
50
60
200
90

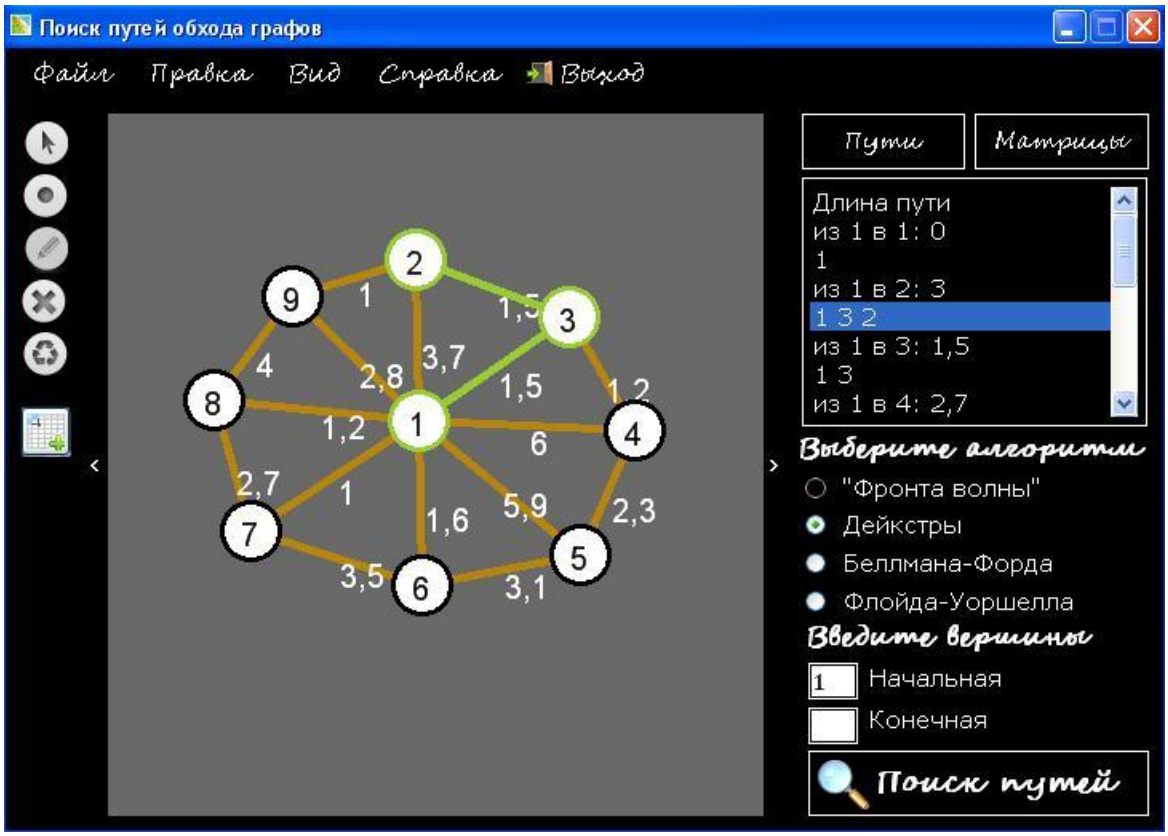

Поиск путей обхода графов
Файл
Правка
Вид
Справка
Выход
Пути
Матрицы
Длина пути:
из 1 в 4: 460
1 5 6 7 9 4
Выберите алгоритм
"Фронта волны"
Дейкстры
Беллмана-Форда
Флойда-Уоршелла
Введите вершины
1
Начальная
4
Конечная
Поиск путей
1
2
3
4
5
6
7
8
9
200
300
150
80
100
50
150
80
100
110
50
60
200
90

Задача 7. Станки в цехе расположены по некоторой схеме и до каждого из них работник может дойти, потратив определенное количество времени.

Требуется определить, какое минимальное количество времени потребуется работнику, чтобы посетить каждый из станков. Начать путь следует с центрального элемента управления, который обозначен 1.

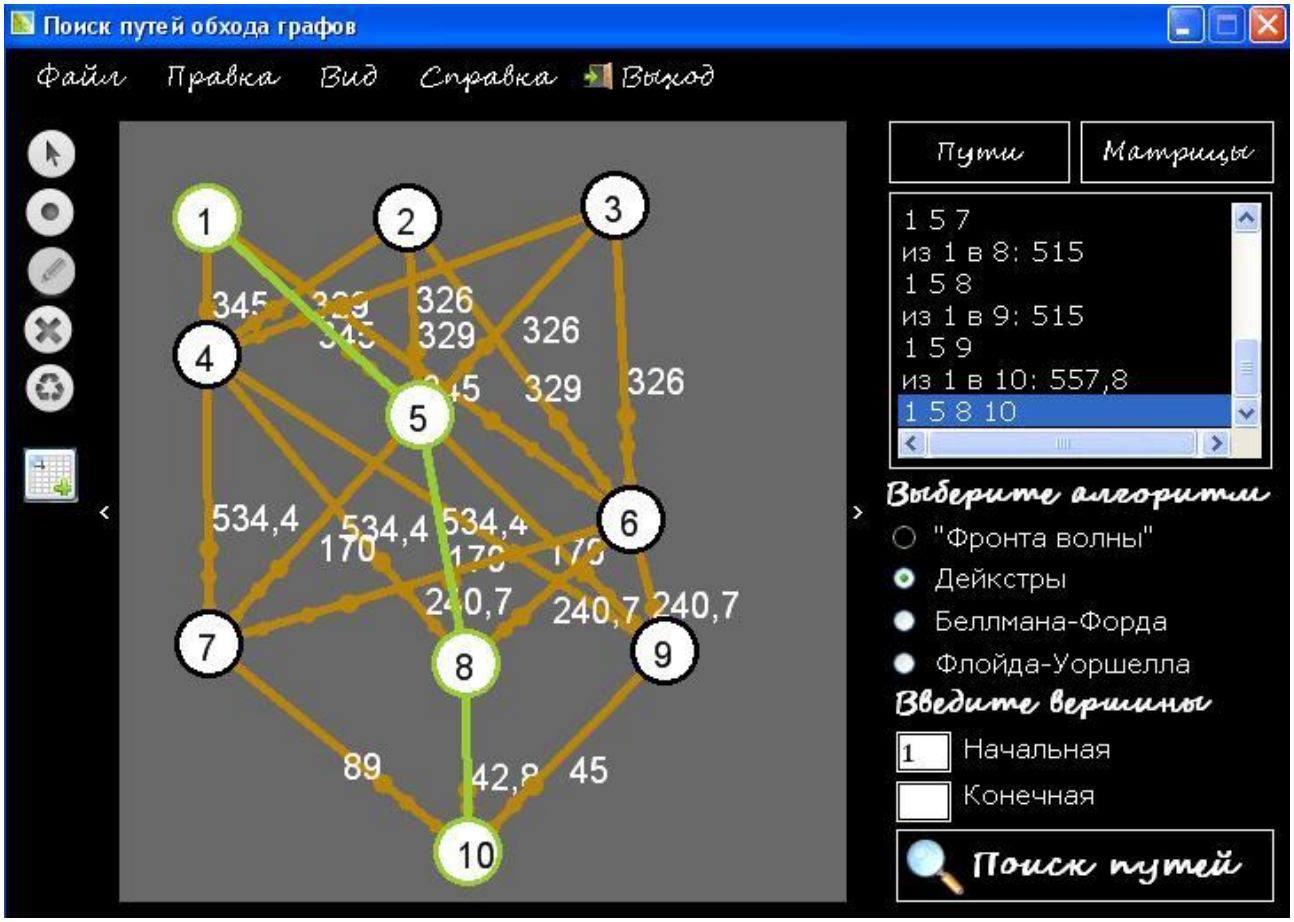


Решение



Задача 8. Требуется составить такой рацион завтрака, который составит наименьшее количество калорий. Рацион обязательно должен состоять из трех продуктов на выбор. 1 категория: каша овсяная (345), каша гречневая (329 ккал) и каша манная (326). 2 категория: бутерброд с сыром (534,4 ккал), йогурт (170 ккал), сухофрукты (240,7 ккал). 3 категория: банан (89 ккал), груша (42,8 ккал), яблоко (45 ккал). Конечную вершину следует принять за состояние сытости

Решение



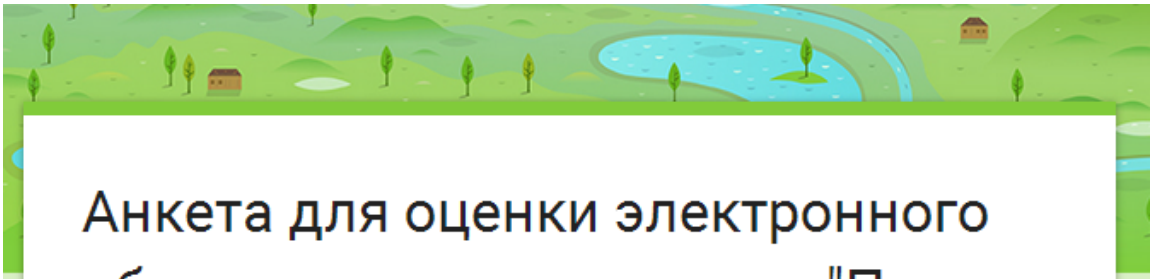
```
//Алгоритм "Фронта волны"
public void Wave()
{   AMatrix = new float[V.Count, V.Count];
    DG.fillAdjacencyMatrix(V.Count, E, AMatrix, MenuOrGr);//Матрица смежности
    int[,] HMatrix;
    HMatrix = new int[V.Count, V.Count];
    string way;
   // string[] n_way;
    int[] N_wave, OldFront, NewFront, n_way;
    N_wave = new int[V.Count];
    int start, finish,T,i,j,row,r,dl,max=1000000,k;
    try
    {
        start = Int32.Parse(textBoxStart.Text) - 1; //стартовая вершина
        finish = Int32.Parse(textBoxFinish.Text) - 1;//конечная вершина
        OldFront = new int[V.Count];//массив для вершин, входящих в новый фронт волны
        NewFront = new int[V.Count];//массив для вершин, входящих в старый фронт волны
        n_way = new int[V.Count];
        way = "";
        T = 0; //счетчик волн
        row = start;
        for (i = 0; i < V.Count; i++)
        {
            N_wave[i] = -1; //массив для записи за какое количество волн мы дошли до каждой из вершин
            OldFront[i] = max;//все элемены массива равны очень большому числу
            NewFront[i] = max;//все элемены массива равны очень большому числу
        }
        N_wave[start] = 0;//исходную вершину мы достигли за 0 волн
        OldFront[0] = start;//начинаем со стартовой вершины
        j = 0;
        dl = 0;
        //до тех пор пока не найдем количество волн для конечной веришны
        while (N_wave[finish] == -1)
        {

    for (k = 0; k < V.Count; k++)
    {
        if (OldFront[k] != max)
        //если вершина с номером k не равна max
        {
            row = OldFront[k];
            for (i = 0; i < V.Count; i++)
                if (AMatrix[row, i] == 1 & N_wave[i] == -1)
                //если вершины смежные и для i не найдено количество волн
                {
                    N_wave[i] = T + 1;//записываем количество волн
                    NewFront[j] = i;//формируем новый фронт
                    j++;
                    dl++;//считаем сколько элементов в NewFront
                }
        }
    }
    if (dl == 0) { listBoxWay.Items.Add("из " + (start + 1) + " в " + (finish + 1) + " путь не существует!")
    for (i = 0; i < V.Count; i++) OldFront[i] = NewFront[i];
    for (i = 0; i < V.Count; i++) NewFront[i] = max;
    j = 0;

    T = T + 1;
}
```

```
//Алгоритм Дейкстры
//Находит пути от каждой вершины до каждой
public void Dijkstry()
{
    int count, index, i, u, start;
    float[] path_length;//длина пути
    bool[] visited; //метка  посещена вершина (true) или нет false
    string[] way; //сам путь
    try {
    visited = new bool[V.Count];
    path_length = new float[V.Count];
    way = new string[V.Count];
    index = 0; //промежуточная переменная
    AMatrix = new float[V.Count, V.Count];
    DG.fillAdjacencyMatrix(V.Count, E, AMatrix, MenuOrGr);//Строится матрица смежности

    start = Int32.Parse(textBoxStart.Text) - 1;//стартовая вершина
    way[start] = (start + 1).ToString() + " "; //строка через которую будет выводится путь
    visited[start] = true;//стартовая вершина посежеа

    for (i = 0; i < V.Count; i++)
    {
        path_length[i] = inf;//заполняется массив с дистанциями большими числами
        visited[i] = false;//отмечаются все вершины как не посещенные
    }

    path_length[start] = 0; //для стартовой вершины дистанция равна 0
    for (count = 0; count < V.Count - 1; count++)
    {

        float min = inf; //берём за минимум бочень большое число
        for (i = 0; i < V.Count; i++)
            //если вершина не посещена и ее дистанция меньше или равна минимуму
            if (!visited[i] && path_length[i] <= min)
      {   //то за минимум берется дистанция этой вершины и запоминаем ее индекс
          min = path_length[i];
          index = i;

      }
    u = index; //запоминается этот индекс
    visited[u] = true;//помечается вершина как пройденная

    for (i = 0; i < V.Count; i++)
    {/*если вершина смежна и не посещена и дистанция до неё не равна максимальному числу
    и расстояние до неё через эту вершину меньше чем ранее просмотренный путь*/
        if (!visited[i] && AMatrix[u, i] != 0 && path_length[u] != inf && path_length[u] + AMatrix[u, i] < path_length[i])
        {
            path_length[i] = path_length[u] + AMatrix[u, i];//то меняем расстояние до нее
            way[i] = String.Concat(way[u], (i+1).ToString()," ");
        }
     }
}

    listBoxWay.Items.Add("Длина пути");
//Вывод результата
for (i = 0; i < V.Count; i++)
        if (path_length[i] != inf)
        {
            listBoxWay.Items.Add("из "+(start + 1) + " в " + (i + 1)+": " + path_length[i].ToString());
            listBoxWay.Items.Add(way[i]);
        }
        else listBoxWay.Items.Add("из " + (start + 1) + " в " + (i + 1) + " нет маршрута!");
}
catch { };
```

```
//Алгоритм Беллмана-Форда
public void Bellman_Ford()
{
    bool x = true;
    int Vmax = 1000;
    int i, j, start, finish,z;
    //int inf = 1000000;

    float[] d;
    int[] path;
    string[] n_way;
    string way;
    d = new float[Vmax]; //массив для длин путей
    path = new int[V.Count]; //массив где для каждой вершины
    //запоминается последняя вершина из которой мы пришли в данную
    try
    {
    start = Int32.Parse(textBoxStart.Text) - 1; //стартовая вершина
    finish = Int32.Parse(textBoxFinish.Text) - 1; //конечная вершина
    way = ""; //строка через которую будет выводится путь
    z = -1;
    for (i = 0; i < V.Count; i++) { d[i] = inf; path[i] = -1; } //все длины равны большому числу, а все пути -1
    d[start] = 0; //для стартовой вершины длина = 0

    for (i = 0; i < V.Count; i++)
    {
        x = false;
        //переменная конроллер, если изменений на последней
        //итерации не произошло, следует прекратить поиск
        for (j = 0; j < E.Count; j++)
            if (d[E[j].startV] < inf || d[E[j].endV] < inf) //если длина стартовой вершины ребра<inf
                //или длина конечно вершины ребра<inf
            {
                if (d[E[j].startV] + E[j].ves < d[E[j].endV]) //если длина стартовой вершины ребра+Вес ребра
                x = true;//изменения были
            }
            //случай для неориентированного графа
            if (d[E[j].endV] + E[j].ves < d[E[j].startV])
            {
                d[E[j].startV] = d[E[j].endV] + E[j].ves;
                path[E[j].startV] = E[j].endV;
                x = true;
            }
        }
    if (!x) { break; } //изменений на итерации не произошло, следует прекратить поиск
}
//вывод пути
    if (d[finish] == inf) listBoxWay.Items.Add("из " + (start + 1) + " в " + (finish + 1) + "путь не существует!");
    else
    {
        for (i = finish; i >= 0; i = path[i])
            way = way + ' ' + (i + 1).ToString(); //формируется путь, возвращаясь по номерам вершин из которых пришли

        n_way = new string[way.Length];
        for (i = 0; i < way.Length; i++)
            n_way[i] = way[i].ToString(); //заносим в массив

        way = "";
        for (i = n_way.Length - 1; i >= 0; i--)
            way = way + n_way[i]; //переворачиваем путь

        listBoxWay.Items.Add("Длина пути:");
        listBoxWay.Items.Add("из " + (start + 1) + " в " + (finish + 1) +": "+d[finish].ToString());
        listBoxWay.Items.Add(way);

    }
```

```
//Алгоритм Флойда-Уоршелла
public void Floid_Uorshell()
{
    int k,i,j;
    AMatrix = new float[V.Count, V.Count];
    int[,] HMatrix;
    HMatrix = new int[V.Count, V.Count];
    DG.fillAdjacencyMatrix(V.Count, E, AMatrix, MenuOrGr); //Создается матрица смежности
    string[,] way;

    try
    {
        way = new string[V.Count, V.Count]; //Массив для хранения путей

            for (i = 0; i < V.Count; i++)
            for (j = 0; j < V.Count; j++)
                if (AMatrix[i, j] == 0) AMatrix[i, j] = inf; //Все элементы кроме диагональных равны inf

        for (i = 0; i < V.Count; i++)
            for (j = 0; j < V.Count; j++)
                if (AMatrix[i, j] != inf) HMatrix[i, j] = j;
                    //в каждом элементе массива пишется номер смежного элемента
                else HMatrix[i, j] = 0;

        for (k = 0; k < V.Count; k++)
            for (i = 0; i < V.Count; i++)
                for (j = 0; j < V.Count; j++)
                    if (AMatrix[i, k] != inf && AMatrix[k, j] != inf && i != j)
                        //если вершины i и K смежны и не равны
                        if (AMatrix[i, k] + AMatrix[k, j] < AMatrix[i, j] || AMatrix[i, j] == 0)
                        //Длина пути от i до j меньше чем длина пути от i до k + длина пути от k до j
                        {
                            AMatrix[i, j] = AMatrix[i, k] + AMatrix[k, j]; //Заменяем длину
                            HMatrix[i, j] = HMatrix[i, k];//Запоминаем что в j пришли из k
                        }
for (i = 0; i < V.Count; i++)
    for (j = 0; j < V.Count; j++)
        if (AMatrix[i, j] != inf) way[i, j] = (i + 1).ToString();
        else way[i, j] = (0).ToString();

//восстановление пути
//проходим по каждой вершине, смотрим из какой вершины в нее пришли и переходим к ней
//таким образом путь восстанавливается по цепочке
for (i = 0; i < V.Count; i++)
    for (j = 0; j < V.Count; j++)
    {
        k = HMatrix[i, j];
        if (HMatrix[i, j] != 0)
        {
            while (k != j)
            {
                way[i, j] = way[i, j] + " " + (k + 1);
                k = HMatrix[k, j];
            }
            way[i, j] = way[i, j] + " " + (k + 1);
        }
    }

listBoxWay.Items.Add("Длина пути:");
for (i = 0; i < V.Count; i++)
    for (j = 0; j < V.Count; j++)
        if (AMatrix[i, j] != inf)
        {
            listBoxWay.Items.Add("из " + (i + 1) + " в " + (j + 1) + ": " + AMatrix[i, j].ToString());
            listBoxWay.Items.Add(way[i, j]);

        }
```

Приложение 4.
Анкета для выявления результатов апробации



Анкета для оценки электронного образовательного ресурса "Поиск пути"

Анкета предназначена для выявления эффективности, удобства интерфейса и качества полученных результатов представленного электронного образовательного ресурса (далее ЭОР).

* Обязательно

На каком курсе вы учитесь? *

- 3 курс
- 4 курс

Если да, то был ли полезен Вам данный ЭОР при решении задач по данному предмету? Если нет, то как Вы предполагаете, мог ли Вам быть полезен данный ЭОР при изучении предмета?

	1	2	3	4	5	
Совсем бесполезен	○	○	○	○	○	Очень полезен

Если Вам понадобилось воспользоваться руководством пользователя или материалами по теории графов при эксплуатации ЭОР, то отыскали ли Вы необходимую информацию?

- Да
- Нет
- Частично

Оцените следующие характеристики представленного Вам ЭОР:

	Да	Скорее да	Затрудняюсь ответить	Скорее нет	Нет
Легко ли Вам было разобраться с тем, как работает данный ЭОР?	○	○	○	○	○
Понятно ли вам назначение панелей, кнопок и текстовых полей ЭОР?	○	○	○	○	○
Понравилась ли Вам цветовая схема, используемая в данном ЭОР?	○	○	○	○	○
Удобно ли расположена панель инструментов, область визуализации графа и панель вывода результатов?	○	○	○	○	○
Удовлетворил ли Вас, полученный результат, был ли он правильным?	○	○	○	○	○
Все ли функции, которые вы ожидали от данного ЭОР, реализованы?	○	○	○	○	○
Понравилось ли Вам работать с данным ЭОР?	○	○	○	○	○
Вызвал ли у Вас данный ЭОР интерес к предмету?	○	○	○	○	○